▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 3

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

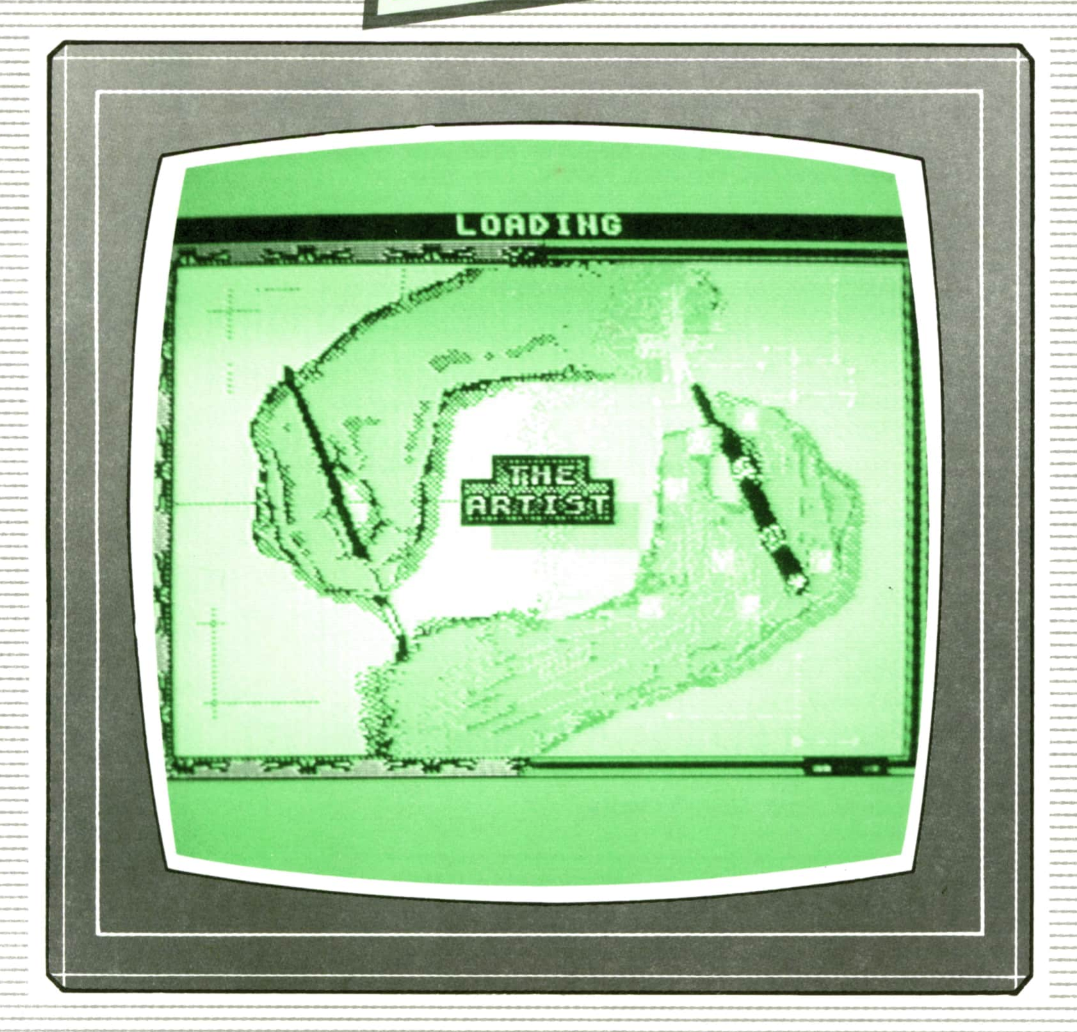
▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación

3

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.

Diseño y maquetación:
BRAVO-LOFISH.

Fotografía:
EQUIPO GALATA.

Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas, Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de Aula de Informática Aplicada (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Todo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: Jesús Bocho, Licenciado en Informática. LOGO: Cristina Manzanero, Licenciada en Informática. APLICACIONES: Fernando Suero, Diplomado en Telecomunicación. OTROS LENGUAJES (Sistemas operativos): Domingo Villaseñor, Diplomado en Informática, y Lenguaje C: Enrique Serrano, Ingeniero en Telecomunicación.

Ediciones Siglo Cultural, S.A.

Dirección, redacción y administración:
Pedro Texeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.

Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-071-5
ISBN de la obra: 84-7688-068-7

Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M. 5.677-1987
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Texeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.

Abril, 1987.
P.V.P. Canarias: 335,–.

INDICE

4 BASIC

7 MAQUINA Z-80

11 PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

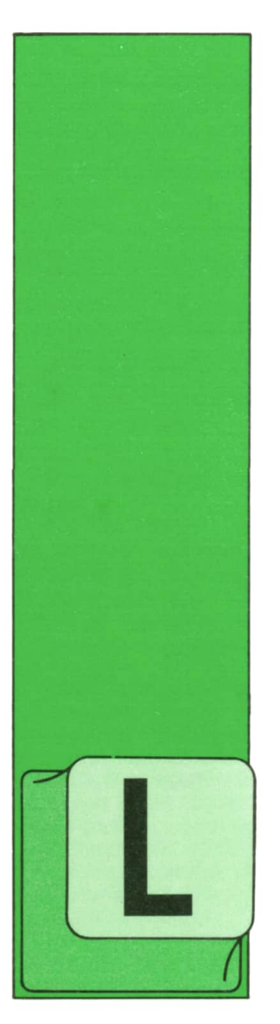
33 TECNICAS DE ANALISIS

35 TECNICAS DE PROGRAMACION

40 LOGO

45 PASCAL

49 OTROS LENGUAJES



BASIC



Las operaciones aritméticas

AS operaciones aritméticas son aquellas que relacionan datos numéricos y son las siguientes:

Nombre	Signo
Suma	+
Resta	−
Multiplicación	*
División	/
Potenciación	^ ó ↑

Podemos realizar estas operaciones con el ordenador, como si fuese una calculadora utilizando PRINT. Si escribimos una operación aritmética a continuación de PRINT, el ordenador nos dará el resultado de dicha operación. Veamos un ejemplo:

```
10 PRINT 12+16
```

Al ejecutar este programa en pantalla aparecerá un 28, resultado de la suma.

Por otra parte, en una expresión aritmética pueden aparecer una o varias operaciones; por tanto, es necesario conocer el orden que sigue el ordenador al efectuar dichas operaciones. Unas operaciones tienen prioridad sobre otras. La prioridad de las operaciones es la siguiente:

1. Potenciación (↑ o ^).
2. Multiplicación (*) y división (/).
3. Suma (+) y resta (-).

En el caso de igualdad de jerarquía (* y / o + y −), las operaciones se efectúan de izquierda a derecha.

Añadamos otra línea a nuestro programa:

```
20 PRINT 24/3−5*2↑3+6↑2
```

Ahora al ejecutar el programa aparecerá un 28 y debajo un 4, resultado de la operación de la línea 20.

El orden seguido por la máquina para efectuar la operación es el siguiente:

24 / 3 − 5 * 2 ↑ 3 + 6 ↑ 2
3.° 5.° 4.° 1.° 6.° 2.°

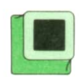
En cualquier caso, siempre es posible alterar la prioridad establecida mediante el uso de los paréntesis necesarios, de modo que si una operación está escrita entre paréntesis se realizará antes que cualquier otra. Por supuesto, si en una operación aparecen varios paréntesis, se realizarán de izquierda a derecha y si hay unos paréntesis dentro de otros, se realizarán de dentro a fuera.

Veamos un ejemplo añadiendo otra línea más a nuestro programa:

```
30 PRINT ((5+3)/2) 2-(3+1)*3+(4-1) 2)
```

Al ejecutar ahora el programa el resultado de esta línea será un 13. El orden seguido en este caso es el siguiente:

((5 + 3) / 2) ↑ 2 - (3 + 1) * 3 + (4 - 1) ↑ 2)
1.° 2.° 5.° 8.° 3.° 7.° 9.° 4.° 6.°



Operaciones con cadenas

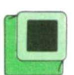
La única operación que podemos realizar con cadenas de caracteres es la concatenación, es decir, podemos formar una cadena única mediante la unión

de dos o más cadenas más sencillas. El signo empleado para la concatenación de cadenas es el de la suma (+), aunque su misión es muy diferente.

Veamos un nuevo programa de ejemplo:

```
10 PRINT "SACA"+"CORCHOS"
20 PRINT "BUENAS"+" "+"NOCHES"
```

Después de ejecutar este programa aparecerá en pantalla la palabra SACACORCHOS y debajo BUENAS NOCHES. El espacio en blanco constituye una cadena más, por lo que va entre comillas.



Novedades en la impresión

Vamos a profundizar más en la instrucción PRINT para poder sacarle un mayor partido. Para conseguir impresiones en pantalla más adecuadas y atractivas resulta bastante útil conocer las dimensiones de la pantalla del ordenador con el que estamos trabajando, es decir, el número de líneas y el número de columnas (número de caracteres por línea). En la figura 1 podemos ver las dimensiones de los principales ordenadores.

	N.º de líneas	N.º de columnas
AMSTRAD	25	20, 40, 80
COMMODORE	25	40
IBM	25	40, 80
MSX	24	32, 40
SPECTRUM	24	32



Dimensiones de pantallas.

El empleo de ciertos separadores (,) o (;) nos permite imprimir varios datos usando una única instrucción PRINT. La coma y el punto y coma tienen diferentes efectos en el aspecto final de la pantalla.

El uso del punto y coma (;) supone que todos los datos incluidos en una misma sentencia PRINT se escribirán con la mínima separación posible entre ellos. Así, al ejecutar el programa siguiente:

```
10 PRINT "OPERACION ";"ARITMETICA"
20 PRINT "18+4=";18+4
```

en pantalla aparecerá la frase OPERACION ARITMETICA y debajo 18 + 4 = 22.

Este separador suele dejar un espacio para el signo en los datos numéricos, teniendo en cuenta que si los números son positivos el signo no se imprimirá, apareciendo en su lugar un espacio en blanco.

Por otra parte, la pantalla del ordenador se suele dividir en zonas verticales, aunque el número de zonas es variable de unas máquinas a otras. En la tabla de la figura 3 podemos ver el número de zonas de los distintos ordenadores en estudio, así como el número de columna de inicio de cada zona.

	N.º de zonas	N.º de columna de inicio de zonas
AMSTRAD	3 6	1, 14, 27 1, 14, 27, 40, 53, 66
COMMODORE	4	0, 10, 20, 30
IBM	2 5	1, 15 1, 15, 29, 43, 57
MSX	2	0, 20 0, 16
SPECTRUM	2	0, 16



Zonas de pantalla.

Visto esto, podemos decir que el uso de la coma (,) para separar distintos datos dentro de una misma instrucción PRINT hace que cada dato aparezca impreso en una zona distinta de la pantalla. Evidentemente, si hay más datos que zonas, éstos continúan imprimiéndose en la fila siguiente, pero siempre respetando las zonas.

Veamos un ejemplo. Supongamos que estamos manejando un ordenador con cuatro zonas de pantalla. Vamos a ejecutar el programa siguiente:

```
10 PRINT "NOMBRE", "APELLIDO", "TFNO.", "EDAD"
20 PRINT "JAVIER", "GARCIA", 9608034,24
```

El resultado en pantalla será algo similar al representado en la figura 4, claro que antes de ejecutar el programa tendremos que teclear el comando CLS para borrar el listado y dejar la pantalla limpia.

NOMBRE JAVIER	APELLIDO GARCIA	TFNO. 960 80 34	EDAD 24



Ejemplo del efecto de la coma.



Funciones y sentencias de impresión

Vamos a analizar a continuación diversas funciones y sentencias que facilitan y hacen más atractiva la impresión de datos en pantalla. Al final de este análisis podemos ver una tabla-resumen de las disponibilidades de distintos ordenadores respecto a ellas:



Función SPC

Esta función permite imprimir en pantalla tantos espacios como deseemos. Su formato es el siguiente:

```
PRINT SPC (N); lista de datos
```

donde *N* es el número de espacios que deseamos imprimir y *lista de datos* son todos los datos de tipo numérico o alfanumérico (separados por comas o puntos y comas) que deseemos imprimir a continuación.

El programa siguiente:

```
10 PRINT SPC (20); "EJEMPLO DE SPC"
```

imprime en pantalla 20 espacios y a continuación la cadena EJEMPLO DE SPC.

En algunos ordenadores la función SPC se puede denominar también SPACE$.



Función TAB

Esta función nos permite realizar impresiones en pantalla a partir de la columna que deseemos. El formato general es el siguiente:

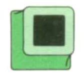
```
PRINT TAB(N); lista de datos
```

donde *N* es el número de columna a partir del cual deseamos imprimir y la *lista de datos* tiene el mismo significado que en SPC.

El programa:

```
10 PRINT TAB(14); "EJEMPLO DE TAB"
```

imprime la cadena EJEMPLO DE TAB a partir de la columna 14 de la pantalla. Si estamos utilizando un ordenador de 40 columnas la cadena aparecerá centrada en la parte superior de la pantalla.



Función AT

La función AT permite la fila y la columna de la pantalla donde queremos realizar la impresión.

El formato es el siguiente:

```
PRINT AT F,C;lista de datos
```

donde *F* es la fila y *C* la columna a partir de las cuales deseamos imprimir la *lista de datos.*

Veamos un ejemplo:

```
10 PRINT AT 12,10; "EJEMPLO DE AT"
```

Este programa imprime la cadena EJEMPLO DE AT en el centro de la pantalla del SPECTRUM.



Sentencia LOCATE

LOCATE es una instrucción independiente de PRINT, aunque ligada a él. Su objetivo es el mismo que AT, es decir, seleccionar la fila y la columna donde se va a realizar la impresión.

Su formato general es el siguiente:

```
LOCATE C,F
```

donde *C* es la columna y *F* es la fila.

Es de observar que los parámetros se escriben en orden inverso a AT, sin embargo, en el IBM el orden es igual que en AT, es decir, primero fila y luego columna.

Veamos un nuevo programa:

```
10 LOCATE 12,13
20 PRINT "EJEMPLO DE LOCATE"
```

La ejecución nos dará como resultado la impresión de la cadena EJEMPLO DE LOCATE en el centro de una pantalla de 25 filas y 40 columnas como, por ejemplo, la del AMSTRAD.

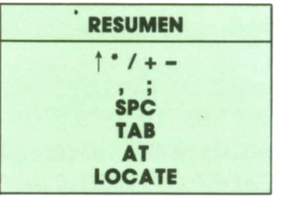
	SPC	TAB	AT	LOCATE
AMSTRAD	•	•		•
COMMODORE	•	•		
IBM	•	•		•
MSX	•	•		•
SPECTRUM		•	•	



Funciones de impresión.



Resumen.



RESUMEN
↑ * / + −
, ;
SPC
TAB
AT
LOCATE

MAQUINA Z-80

MODOS DE DIRECCIONAMIENTO



A mayoría de las instrucciones en lenguaje máquina necesitan utilizar datos. Estos pueden representar números de 8 bits (Bytes) o de 16 (Words) y pueden encontrarse en registros de la propia CPU o en memoria.

De la forma en cómo le indicamos al microprocesador dónde encontrar estos datos tratan los modos de direccionamiento. Intentaremos explicar al lector en qué consiste cada uno de los disponibles en el Z-80. Estos son:

— Implícito.

— Inmediato.

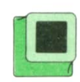
— De registros.

— Directo.

— Indirecto.

— Relativo.

— Indexado.



Direccionamiento implícito

Esta forma de direccionamiento es la utilizada en aquellas instrucciones en las que la misma instrucción indica dónde se halla el dato a utilizar de forma implícita. En la figura 1 puede verse el flujo de datos típico de esta forma de direccionar.

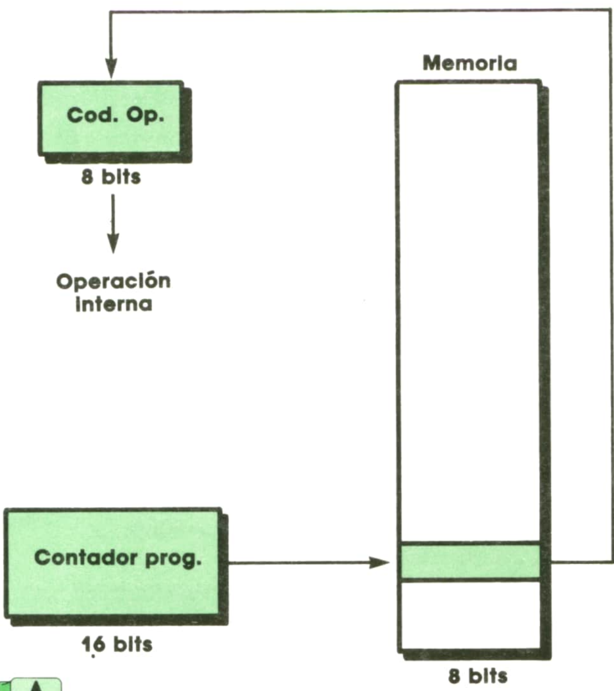




Diagrama direccionamiento implícito.

Ejemplos de instrucciones con este tipo de direccionamiento son:

— SFC (Set Carry Flag): Pone a 1 la bandera (flag), C, de acarreo. Afecta al registro F de forma *implícita.*

— CPL: Complementa el acumulador (cambia ceros por unos, y viceversa). (A → A). Evidentemente, afecta de forma implícita al acumulador.



Direccionamiento inmediato

En esta forma de direccionamiento, el dato a utilizar es un número que sigue a

la instrucción. En la figura 2 puede verse el flujo de datos. Con este sistema no es necesaria ninguna dirección, ya que el dato va incluido de forma numérica en la misma instrucción. Por ejemplo:

LD A,$2F

— El signo $ indica que el número va en notación hexadecimal.

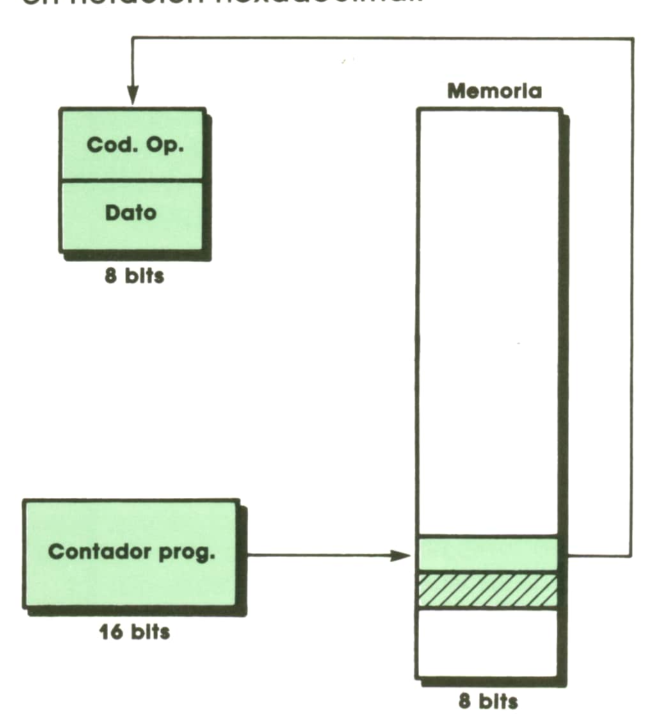
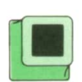



Diagrama direccionamiento inmediato.

— LD: (LoaD) quiere decir cargar.
— A; acumulador.

Por tanto, esta instrucción carga en el acumulador el dato que le sigue *inmediatamente* después, es decir, $2F.



Direccionamiento de registros

Con esta forma, en principio, el propio código de operación de la instrucción indica cuál es el registro en el que se encuentra el dato a utilizar y su destino, como puede verse en la figura 3.

Por ejemplo:

— LD B,C carga en el registro B el contenido del C.

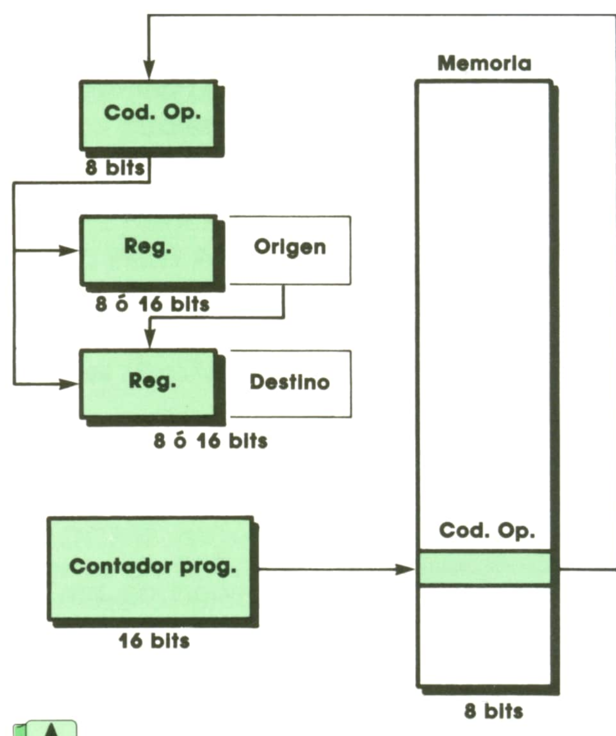



Diagrama direccionamiento de registros.

— EX DE,AL intercambia los contenidos de los pares de 16 bits DE y AL.

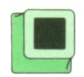
— ADD A,B suma los contenidos de los registros A y B.

Este tipo de direccionamiento suele encontrarse combinado con otros tipos de direccionamiento, como el inmediato.

Por ejemplo: LD A,$16

Carga en A el número $16 que es inmediato.

Esta forma de combinar dos modos de direccionamiento es bastante habitual.



Direccionamiento directo

Este modo de direccionamiento es el más utilizado para tomar o dejar datos de la memoria, ya que los anteriores no tomaban la dirección de la memoria principal.

Este tipo ocupa 3 bytes, en el que el primer byte indica la operación a realizar y los dos siguientes juntos forman una dirección de 16 bits de la memoria principal.

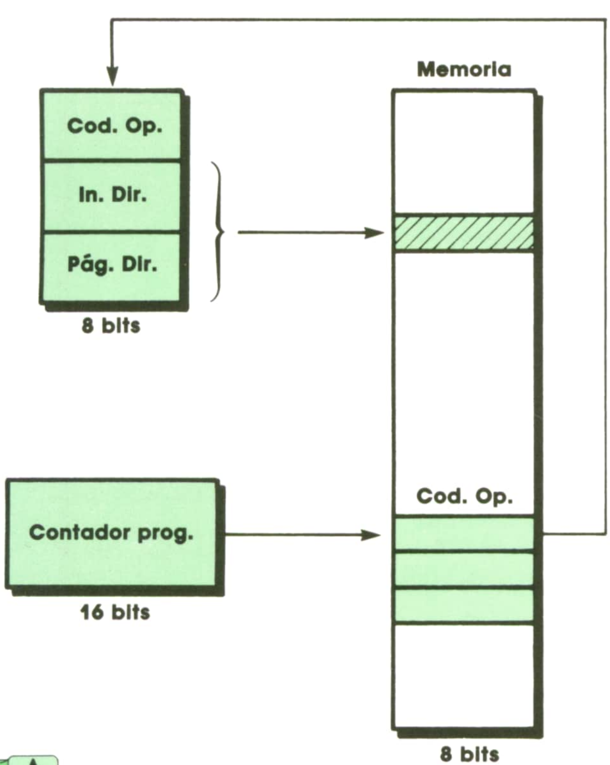




Diagrama direccionamiento directo.

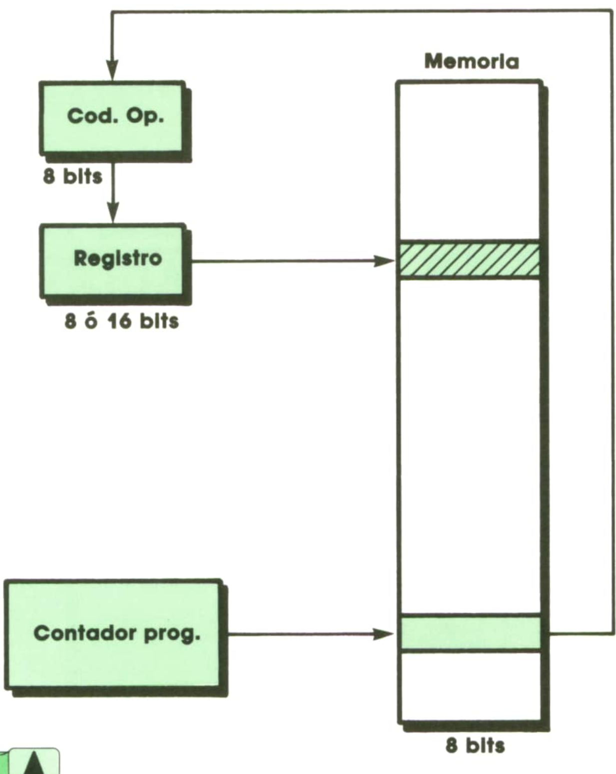




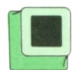
Diagrama direccionamiento indirecto.

Presenta la ventaja de ser de un uso muy general, pero utiliza mucho más tiempo que los anteriores, ya que tiene que acceder a la memoria principal para determinar la dirección efectiva. Además, ocupa más memoria.

Un ejemplo de uso de esta instrucción sería:

LD A($1348)

Esta instrucción cargaría el contenido de la posición $1348 en el acumulador.



Direccionamiento indirecto

En este tipo no se indica la dirección en la que está el dato, sino la dirección del registro donde está contenida la dirección de memoria en la que está el dato. Esto puede parecer un poco complicado a primera vista, pero no lo es tanto observando la siguiente figura.

Las ventajas de este tipo respecto del directo es que ocupa menos memoria y tarda menos tiempo, pero tiene el defecto de que la dirección debemos cargarla anteriormente en el registro a utilizar indirectamente.

Un ejemplo sería:

LD A,(B,C)

que cargaría el acumulador con la dirección indicada por el contenido de los registros B y C.

Direccionamiento relativo

En este caso la dirección efectiva se obtiene al sumar un desplazamiento (de -127 a 128) a una dirección predeterminada.

Dicho desplazamiento viene indicado por un byte que sigue a la instrucción.

La dirección predeterminada puede ser el contenido del contador de programa (dirección de la instrucción que está ejecutando).

Otra posibilidad es que la predirección esté en un registro. Para esta función están diseñados los registros de indirección IX e IY.

Un ejemplo sería:

LD A(IX + $9)

que cargaría el acumulador con el contenido de la posición que esté 9 bytes por delante de la indicada en el registro IX.

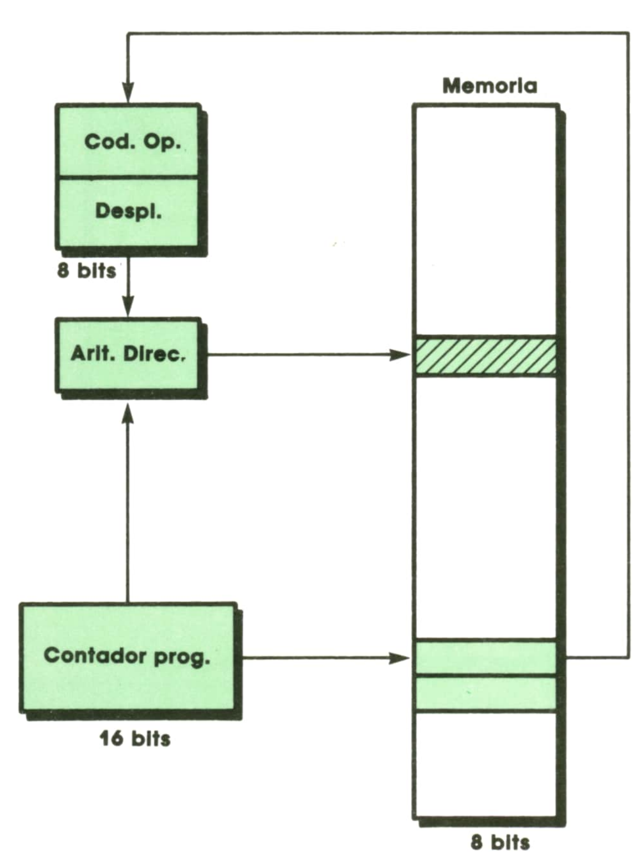



Diagrama direccionamiento relativo.



Direccionamiento indexado

Este es una variante del anterior, con la particularidad que tras la ejecución el registro índice utilizado se ve incrementado en uno, con lo que está preparado para repetir el mismo tipo de direccionamiento y seleccionar la siguiente posición de memoria.

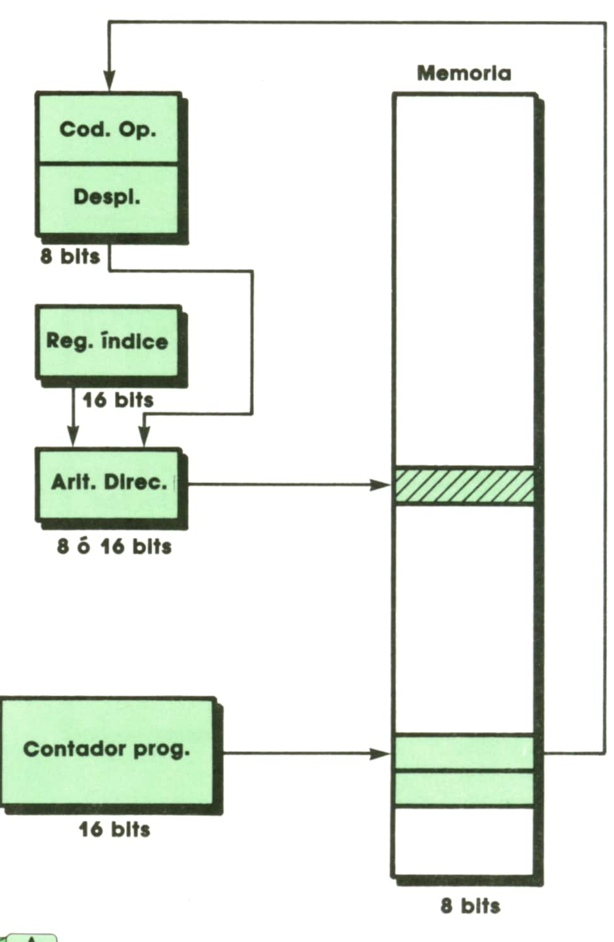



Diagrama direccionamiento indexado.

Este tipo y el anterior son especialmente indicados para trabajar con tablas de datos y matrices.

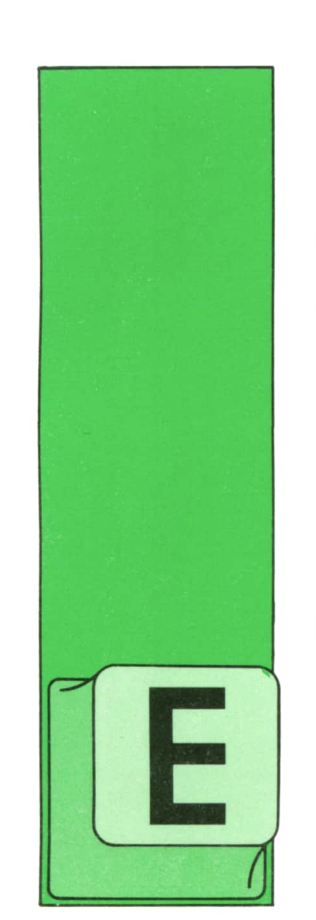
Un ejemplo sería:

LDI A,(BC)

que cargaría la dirección indicada por el contenido de BC en el acumulador e incrementaría el contenido de BC en uno.

Con esto ya conocemos todos los tipos de direccionamiento existentes en el Z-80.

Por ello, el siguiente paso será ver detalladamente todas las instrucciones que utilizarán estos direccionamientos, viendo su utilidad práctica.



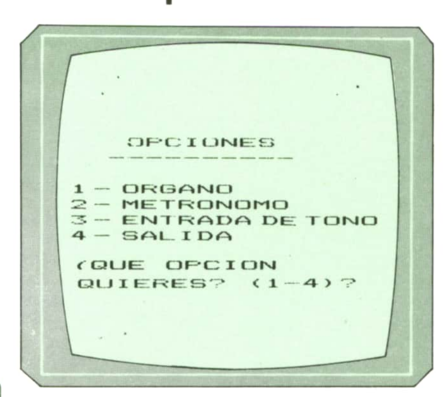
PROGRAMAS

PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS



Programa: Organo electrónico para IBM

ESTE primer programa de este tercer tomo es un órgano electrónico que nos permitirá componer, interpretar, modificar y escuchar todo tipo de música y ritmos a una sola voz. Aunque este programa, funcione sólo en el IBM PC, XT y AT, así como en todos los compatibles, aparecerán en otros tomos las versiones para SPECTRUM, AMSTRAD y MSX.





Menú general del programa órgano para IBM.

```
100 REM ***********************************
101 REM *                                 *
102 REM *        MAQUINA DE SONIDO        *
103 REM *                                 *
104 REM * VARIABLES:                      *
105 REM *                                 *
106 REM * NP   - NUMERAL DE PARTE (OPT)   *
107 REM * TP   - TEMPO (ORG, MET Y INPUT) *
108 REM * N$   - NOTA   (ORG Y INPUT)     *
109 REM * LB$ - LABEL DE KEY (ORG)        *
110 REM * HZ   - FREQUNCIA (MET)          *
111 REM * DR1 - DURATION DE NOTA (MET)    *
112 REM * DR2 - DURATION DE REST (MET)    *
113 REM * FL$ - FLAG DE ALTO (MET)        *
114 REM *      - FLAG DE NOTA MALA (INPUT)*
115 REM * X,Y - COORDENADAS DE CIRC.(MET)*
116 REM * X1,2- LO MISMO                  *
117 REM * NN$ - ETIQUETA DE NOTA (ORG)    *
118 REM * AN$ - ARRAY DE NOTAS (INPUT)    *
119 REM * RN   - NUMERAL DE FILA (INPUT)  *
120 REM * CN   - NUMERAL DE COLUMN (INPUT)*
121 REM * AL$ - FLAG DE INPUT  (INPUT)    *
122 REM * N    - NUMERAL DE ELEMENTE(INP) *
123 REM * S1$ - STRING DE MUSICA (INPUT) *
124 REM * S2$ - LO MISMO                  *
125 REM * BC$ - NOTA BUENA (INPUT)        *
```

```
126 REM * R$   - RESPUESTA (SI O NO)       *
127 REM * C$   - RESPUESTA                 *
128 REM * K,L,I - INDICES                  *
129 REM *                                  *
130 REM * LAS PALABRAS ENTRE COMILLAS SON*
131 REM * LAS DIFERENTES PARTES DEL PRO- *
132 REM * GRAMA.                           *
133 REM **********************************
134 REM
135 KEY OFF
136 REM
137 REM **************
138 REM *  CABECERA  *
139 REM **************
140 REM
141 CLS
142 WIDTH 40
143 PRINT "****************************************"
144 FOR I = 1 TO 19
145    PRINT  "*"; TAB(40); "*"
146 NEXT I
147 PRINT "****************************************"
148 LOCATE 8,13: PRINT "ORGANO ELECTRICO"
149 LOCATE 9,12: PRINT "------------------"
150 LOCATE 12,7: PRINT "(c) Ed. Siglo Cultural, 1987"
151 FOR I = 1 TO 3000
152 NEXT I
153 REM**************************
154 REM*       ELEGIR OPCION      *
155 REM**************************
156 CLS
157 SCREEN 0,0,0
158 LOCATE 3,16: PRINT "OPCIONES"
159 LOCATE 4,15: PRINT "----------"
160 LOCATE 6,13: PRINT "1  -  ORGANO"
161 LOCATE 7,13: PRINT "2  -  METRONOMO"
162 LOCATE 8,13: PRINT "3  -  ENTRADA DE TONO"
163 LOCATE 9,13: PRINT "4  -  SALIDA"
164 LOCATE 13,3: PRINT "¿QUE OPCION QUIERES? (1-4)";
165 INPUT NP
166 ON NP GOSUB 170,286,340,168
167 GOTO  155
168 KEY ON
169 END
170 REM********************************
171 REM*            ORGANO            *
172 REM********************************
173 CLS
174 SCREEN 0,0,0
175 LOCATE 4,3: PRINT "PULSA 'S' PARA TERMINAR"
176 LOCATE 5,3: PRINT "PULSA 'N' PARA NUEVO TEMPO"
177 PRINT:PRINT "INPUT TEMPO (32-255) - ";
178 INPUT TP
179 GOSUB 216
180 PLAY "T=" + VARPTR$(TP)
181 PLAY "ML MB"
182 N$ = INPUT$ (1)
183 IF N$ = "Q" THEN PLAY"O1C8":RESTORE 265: GOSUB 205
184 IF N$ = "2" THEN PLAY"O1C#8":RESTORE 266: GOSUB 205
185 IF N$ = "W" THEN PLAY"O1D8":RESTORE 267: GOSUB 205
186 IF N$ = "3" THEN PLAY"O1D#8":RESTORE 268: GOSUB 205
187 IF N$ = "E" THEN PLAY"O1E8":RESTORE 269: GOSUB 205
188 IF N$ = "R" THEN PLAY"O1F8":RESTORE 270: GOSUB 205
189 IF N$ = "5" THEN PLAY"O1F#8":RESTORE 271: GOSUB 205
190 IF N$ = "T" THEN PLAY"O1G8":RESTORE 272: GOSUB 205
191 IF N$ = "6" THEN PLAY"O1G#8":RESTORE 273: GOSUB 205
192 IF N$ = "Y" THEN PLAY"O1A8":RESTORE 274: GOSUB 205
```

```
193 IF N$ = "7" THEN PLAY"O1A#8":RESTORE 275: GOSUB 205
194 IF N$ = "U" THEN PLAY"O1B8":RESTORE 275: GOSUB 205
195 IF N$ = "I" THEN PLAY"O2C8":RESTORE 277: GOSUB 205
196 IF N$ = "9" THEN PLAY"O2C#8":RESTORE 278: GOSUB 205
197 IF N$ = "O" THEN PLAY"O2D8":RESTORE 279: GOSUB 205
198 IF N$ = "0" THEN PLAY"O2D#8":RESTORE 280: GOSUB 205
199 IF N$ = "P" THEN PLAY"O2E8":RESTORE 281: GOSUB 205
200 IF N$ = "N" THEN GOTO 173
201 IF N$ = "S" THEN GOTO 203
202 GOTO 182
203 SCREEN 0,0,0
204 RETURN
205 REM
206 REM ** BLINK KEY **
207 REM
208 READ X,Y,LB$
209 LOCATE X,Y
210 PRINT CHR$(219)
211 FOR I = 1 TO 100
212 NEXT I
213 LOCATE X,Y
214 PRINT LB$
215 RETURN
216 REM
217 REM   *****  DIBUJA COSAS   *****
218 REM
219 CLS
220 SCREEN 1,1
221 REM
222 REM   ****  BORDE  ****
223 REM
224 LINE (5,100) - (305,100)
225 LINE (5,5) - (305,5)
226 LINE (5,5) - (5,100)
227 LINE (305,5) - (305,100)
228 REM
229 REM   ****  TECLAS BLANCAS ****
230 REM
231 LET X = 35
232 FOR I = 1 TO 9
233 IF (I=3) OR (I=7) THEN Y=5 ELSE Y=70
234 LINE (X,Y) - (X,100)
235 X = X + 30
236 NEXT I
237 REM
238 REM   **** TECLAS NEGRAS ****
239 REM
240 LET X1 = 28
241 FOR I = 1 TO 9
242 IF (I=3) OR (I=7) GOTO 247
243 LINE (X1,5) - (X1,70)
244 X2 = X1 + 14
245 LINE (X2,5) - (X2,70)
246 LINE (X1,70) - (X2,70)
247 X1 = X1 + 30
248 NEXT I
249 REM
250 REM   **** ETIQUETAS DE LAS TECLAS ****
251 REM
252 RESTORE
253 FOR I = 1 TO 17
254 READ X,Y,LB$
255 LOCATE X,Y: PRINT LB$
256 NEXT I
257 REM
258 REM  **** ETIQUETAS CON NOMBRE ****
```

```
259 REM
260 FOR I = 1 TO 10
261 READ Y,NN$
262 LOCATE 14,Y: PRINT NN$
263 NEXT I
264 RETURN
265 DATA 11,3,"Q"
266 DATA 8,5,"2"
267 DATA 11,7,"W"
268 DATA 8,9,"3"
269 DATA 11,11,"E"
270 DATA 11,14,"R"
271 DATA 8,16,"5"
272 DATA 11,18,"T"
273 DATA 8,20,"6"
274 DATA 11,22,"Y"
275 DATA 8,24,"7"
276 DATA 11,25,"U"
277 DATA 11,29,"I"
278 DATA 8,31,"9"
279 DATA 11,33,"O"
280 DATA 8,35,"0"
281 DATA 11,37,"P"
282 DATA 3,"C",7,"D",11,"E",14,"F",18
283 DATA "G",22,"A",25,"B",29,"C",33
284 DATA "D",37,"E"
285 REM
286 REM*********************************
287 REM*           METRONOMO           *
288 REM*********************************
289 REM
290 ON KEY(1) GOSUB 336
291 ON KEY(11) GOSUB 321
292 ON KEY(14) GOSUB 325
293 ON KEY(13) GOSUB 329
294 ON KEY(12) GOSUB 333
295 CLS
296 PRINT "PULSA 'F1' PARA TERMINAR"
297 PRINT:PRINT "DAME LA FRECUENCIA (40-8000) - ";:INPUT HZ
298 IF (HZ > 8000) OR (HZ < 40) GOTO 297
299 PRINT:PRINT "DAME LA DURACION (1-5) - ";:INPUT DR1
300 IF (DR1 < 1) OR (DR1 > 5) GOTO 299
301 PRINT "DAME LA PAUSA (1-25) - ";:INPUT DR2
302 IF (DR2 < 1) OR (DR2 > 25) GOTO 301
303 LET N = 0
304 SCREEN 1,1
305 LET FL$ = "Y"
306 KEY(1) ON
307 KEY(11) STOP: KEY(14) STOP
308 KEY(12) STOP: KEY(13) STOP
309 CLS
310 SOUND HZ, DR1
311 X = INT(159 + 90*(COS(N)))
312 Y = INT(99 + 90*(SIN(N)))
313 CIRCLE ( X, Y),5,2
314 N = N + .1
315 KEY(11) ON: KEY(14) ON
316 SOUND 0, DR2
317 KEY(12) ON: KEY(13) ON
318 IF FL$ = "Y" GOTO 307
319 RETURN
320 REM ** AUMENTA LA FRECUENCIA **
321 IF HZ > 8000 GOTO 323
322 HZ = HZ + 25
323 RETURN
324 REM ** DISMINUYE LA FRECUENCIA **
```

```
325 IF HZ <= 65 GOTO 327
326 HZ = HZ - 25
327 RETURN
328 REM ** DISMINUYE LA DURACION DE LA PAUSA **
329 IF DR2 <= 1 GOTO 331
330 DR2 = DR2 - .5
331 RETURN
332 REM ** AUMENTA LA DURACION DE LA PAUSA **
333 DR2 = DR2 + .5
334 RETURN
335 REM
336 LET FL$ = "N"
337 RETURN
338 REM
339 REM
340 REM**********************************
341 REM*       EDITOR DE MUSICA          *
342 REM**********************************
343 REM
344 CLS
345 PRINT "(QUIERES INSTRUCCIONES? (S/N)";
346 INPUT R$
347 IF R$ = "S" GOTO 350
348 IF R$ = "N" GOTO 396
349 GOTO 345
350 REM  ** INSTRUCCIONES **
351 REM
352 CLS
353 LOCATE 3,13: PRINT "INSTRUCCIONES"
354 LOCATE 4,12: PRINT "---------------"
355 PRINT
356 PRINT "  PARA ESCRIBIR MUSICA NECESITAS DERLE"
357 PRINT "LA PROGRAMMA TRES COSAS:":PRINT
358 PRINT "         1  -  QUE OCTAVA"
359 PRINT "         2  -  QUE NOTA"
360 PRINT "         3  -  QUE LONGITUD"
361 PRINT
362 PRINT "LAS OCTAVAS SE ESCRIBEN CON 'On'"
363 PRINT " - n ES LA NUMERO DE LA OCTAVA (0-6)."
364 PRINT
365 PRINT "LAS NOTAS SON 'ABCDEFG' CON '#' Y '-'."
366 PRINT
367 PRINT "LAS LONGITUDES SON '1,2,3,4,8 ETC."
368 PRINT
369 PRINT "PAUSAS SON 'P' SEGUIDO DE '1,2,4,8'"
370 PRINT
371 PRINT "(ENTER PARA CONTINUAR)"
372 INPUT C$
373 CLS
374 PRINT
375 PRINT "POR EXEMPLO, LA SIGUENTE LINEA TOCA UNA "
376 PRINT "'A' EN LA OCTAVA 1, UNA 'B' EN LA OCTAVA 2,"
377 PRINT "Y UNA 'C' EN LA OCTAVA 3. FIJATE QUE LA ULTIMA"
378 PRINT "NOTA ES MAS LARGA."
379 PRINT
380 PRINT
381 PRINT "O1A4 O2B4 O3C1"
382 PRINT
383 PRINT "(RETURN ESCUCHAR)"
384 INPUT R$
385 PLAY "MF T100 O1A4 O2B4 O3C1"
386 PRINT
387 PRINT
388 PRINT "UNA 'L' SEGUIDA DE UN NUMERO HACE QUE TODAS"
389 PRINT "LAS NOTAS TENGAN LA MISMA LONGITUD":PRINT
390 PRINT "FIJATE EN EL EJEMPLO ANTERIOR"
```

```
391 PRINT
392 PRINT "L4 O1A O2B O3C2"
393 PRINT
394 PRINT "(ENTER PARA CONTINUAR)"
395 INPUT R$
396 CLS
397 REM
398 REM ** ENTRADA DEL TONO **
399 REM
400 DIM AN$(360)
401 FOR I = 1 TO 360
402 LET AN$(I) = " "
403 NEXT I
404 CLS
405 PRINT "INTRODUCE LAS NOTAS  (X PARA FIN): "
406 LET I = 1
407 LET N$ = " "
408 WHILE (N$ <> "X") AND (I <= 360)
409 N$ = INPUT$(1)
410 GOSUB 511
411 IF FL$ = "N" THEN GOTO 408
412 PRINT N$;
413 AN$(I) = N$
414 I = I + 1
415 WEND
416 PRINT
417 PRINT
418 IF I > 360 THEN LOCATE 10,5: PRINT "NUMERO MAXIMO DE NOTAS AGOTADO"
:PRINT "P ULSA RETURN.";: INPUT R$
419 REM
420 REM ** EDITAR CANCION **
421 REM
422 CLS
423 LOCATE 1,1: PRINT "(QUIERES CAMBIAR LA MUSICA? (S/N)";
424 INPUT R$
425 IF R$ = "N" GOTO 469
426 IF R$ <> "S" GOTO 423
427 CLS
428 PRINT "USA <- Y -> PARA MOVER EL CURSOR": PRINT "USA 'F1' PARA ENTRAR
UNA NOTA NUEVA" :PRINT "USA 'F1' Y 'X' A LA VEZ PARA SALIR"
429 GOSUB 431
430 GOTO 449
431 REM
432 REM ** IMPRESION DE LA CADENA MUSICAL **
433 REM
434 LET RN = 3
435 LET CN = 1
436 LET N = 1
437 FOR K = 1 TO 9
438 RN = RN + 2
439 LOCATE RN, CN
440 FOR L = 1 TO 40
441 PRINT AN$ (N);
442 N = N + 1
443 NEXT L
444 NEXT K
445 LET RN = 6: LET CN = 1
446 LOCATE RN, CN: PRINT "-"
447 AL$ = "N"
448 RETURN
449 REM
450 REM ** RUTINAS DE MOVIMIENTO DEL CURSOR **
451 REM
452 ON KEY(13) GOSUB 523
453 ON KEY(1) GOSUB 539
454 ON KEY(12) GOSUB 531
```

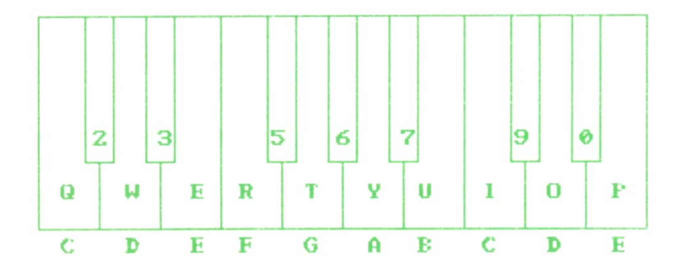
```
455 KEY(12) ON: KEY(13) ON: KEY(1) ON
456 IF AL$ = "N" GOTO 452
457 KEY(12) OFF: KEY(13) OFF: KEY(1) OFF
458 N$ = INPUT$(1)
459 GOSUB 511
460 IF N$ = "X" GOTO 419
461 N = (RN - 6)*20 + CN
462 LET AN$(N) = N$
463 LOCATE RN-1, CN: PRINT N$
464 LOCATE RN, CN: PRINT " "
465 CN = CN + 1
466 LOCATE RN, CN: PRINT "-"
467 LET AL$ = "N"
468 GOTO 452
469 REM
470 REM ** PLAY STRING **
471 REM
472 REM
473 REM  ** ENSAMBLAJE **
474 REM
475 S1$ = ""
476 FOR J = 1 TO 180
477 S1$ = S1$ + AN$ (J)
478 NEXT J
479 S2$ = ""
480 FOR J = 181 TO 360
481 S2$ = S2$ + AN$ (J)
482 NEXT J
483 CLS
484 PRINT "¿QUE TEMPO QUIERES? (32-255):";
485 INPUT TP
486 IF (TP < 32) OR (TP > 255) GOTO  483
487 GOSUB 432
488 PLAY "MF T=" + VARPTR$ (TP)
489 ON ERROR GOTO 541
490 PLAY "X" + VARPTR$(S1$)
491 ON ERROR GOTO 541
492 PLAY "X" + VARPTR$(S2$)
493 CLS
494 LOCATE 7,1: PRINT "¿QUIERES CAMBIAR ESTA CANCION? (S/N)";
495 INPUT R$
496 IF R$ = "N" GOTO 499
497 IF R$ <> "S" GOTO 494
498 GOTO  427
499 LOCATE 8,1: PRINT "¿QUIERES TOCAR ESTA CANCION? (S/N)";
500 INPUT R$
501 IF R$ = "N" GOTO 504
502 IF R$ <> "S" GOTO 499
503 GOTO 469
504 LOCATE 9,1: PRINT "¿QUIERES COMPONER OTRA CANCION: (S/N):";
505 INPUT R$
506 IF R$ = "N" GOTO 509
507 IF R$ <> "S" GOTO 504
508 GOTO 401
509 RETURN
510 END
511 REM
512 REM
513 REM
514 RESTORE 522
515 LET FL$ = "N"
516 FOR K = 1 TO 18
517 READ BC$
518 IF BC$ = N$ THEN FL$ = "S"
519 NEXT K
520 IF (FL$ = "N") AND (N$ <> "X") THEN SOUND 100, 3
```

```
521 RETURN
522 DATA "L","O","P","A","B","C","D","E","F","G","1","2","3","4","8"," ","#","-"
523 REM
524 REM ** MOVIMIENTO DEL CURSOR A LA DERECHA **
525 REM
526 IF (CN = 40) AND (RN = 22) GOTO 530
527 LOCATE RN, CN: PRINT " "
528 IF CN < 40 THEN CN = CN + 1 ELSE LET CN = 1 : RN = RN + 2
529 LOCATE RN, CN: PRINT "-"
530 RETURN
531 REM
532 REM ** MOVIMIENTO DEL CURSOR A LA IZQUIERDA **
533 REM
534 LOCATE RN, CN: PRINT " "
535 IF (CN = 1) AND (RN > 6) THEN LET CN = 40: RN = RN - 2
536 IF CN > 1 THEN CN = CN - 1
537 LOCATE RN, CN: PRINT "-"
538 RETURN
539 AL$ = "Y"
540 RETURN
541 REM
542 REM ** ERROREN EL STRING **
543 CLS
544 REM
545 PRINT "HAY ERROR EN LA MUSICA."
546 PRINT "PUEDE SER POR :":PRINT
547 PRINT "  1 - UNA 'O' SIN NUMERO DESPUES"
548 PRINT "  2 - UNA 'L' SIN NUMERO DESPUES"
549 PRINT "  3 - UNA 'P' SIN NUMERO DESPUES"
550 PRINT:PRINT "COMPRUEBA QUE TODAS LAS O's, L's Y P'S"
551 PRINT "VAN SEGUIDAS DE UN NUMERO."
552 RESUME 494
```

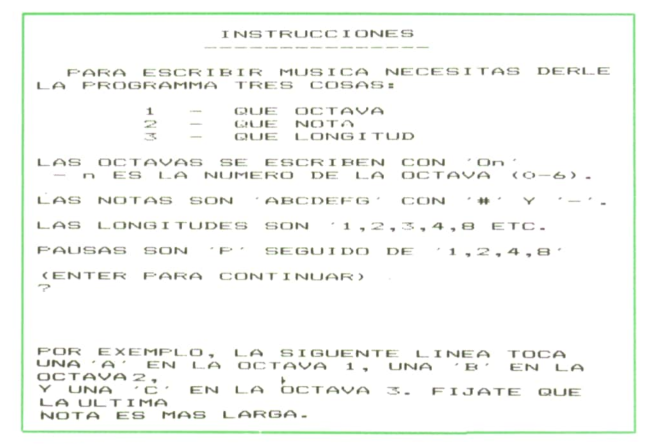
Este programa se divide entre partes distintas:

1. El órgano propiamente dicho. En ella aparece el dibujo de un teclado musical con las teclas que tenemos que pulsar para que suene la música. Antes de aparecer el dibujo del teclado, el programa nos preguntará el *tempo* y nos dará un par de mensajes de información por pantalla.

2. El metrónomo es un aparato que utilizan los músicos para no perder el ritmo. Esta parte del programa simula uno de ellos. Antes de empezar dicha simulación, el programa nos pedirá la frecuencia del sonido, la duración y la pausa entre sonido y sonido.




Así aparece el teclado del órgano en la pantalla del ordenador.



INSTRUCCIONES

PARA ESCRIBIR MUSICA NECESITAS DERLE LA PROGRAMMA TRES COSAS:

1 - QUE OCTAVA
2 - QUE NOTA
3 - QUE LONGITUD

LAS OCTAVAS SE ESCRIBEN CON 'On' - n ES LA NUMERO DE LA OCTAVA (0-6).

LAS NOTAS SON 'ABCDEFG' CON '#' Y '-'.

LAS LONGITUDES SON '1,2,3,4,8 ETC.

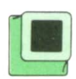
PAUSAS SON 'P' SEGUIDO DE '1,2,4,8'

(ENTER PARA CONTINUAR)
?

POR EXEMPLO, LA SIGUENTE LINEA TOCA UNA 'A' EN LA OCTAVA 1, UNA 'B' EN LA OCTAVA 2, Y UNA 'C' EN LA OCTAVA 3. FIJATE QUE LA ULTIMA NOTA ES MAS LARGA.


Instrucciones de uso del programa «Organo».

3. Un compositor musical. Con él podemos componer música para que la toque el ordenador o bien introducir alguna melodía que conozcamos. Incluso nos permite modificar la melodía si nos hemos equivocado.



Programa: Dragones y mazmorras

Este programa es un juego que nos introduce dentro de las mazmorras de un lóbrego castillo medieval. Dentro del castillo tenemos que encontrar 4 llaves que nos permitirán encontrar al amo del calabozo.

```
 10 BORDER 7: PAPER 7: INK 7: B
RIGHT 0: OVER 0: INVERSE 0: CLEA
R 64567
 20 PRINT  INK 1; PAPER 7; BRIG
HT 1; FLASH 1;AT 10,6;"DRAGONES
Y MAZMORRAS"
 30 LOAD ""CODE
 40 LOAD ""
```

```
  3 GO TO 9600
  5 GO TO 7000
 10 REM Paredes de pantallas
 20 LET l=USR 65153: FOR f=1 TO
 7: PRINT  INK 3; BRIGHT 1;AT f,
0;"P": NEXT f: FOR f=8 TO 12: PR
INT  INK 1; BRIGHT 0;AT f,0;" ":
 NEXT f: FOR f=13 TO 19: PRINT
INK 3; BRIGHT 1;AT f,0;"P": NEXT
 f: LET l=USR 64958: PRINT  INK
3; BRIGHT 1;AT 0,0;"C";AT 20,0;"
B"
 23 RETURN
 25 LET l=USR 65153: FOR f=1 TO
 7: PRINT  INK 3; BRIGHT 1;AT f,
31;"Q": NEXT f: FOR f=8 TO 12: P
RINT  INK 1; BRIGHT 0;AT f,31;"
": NEXT f: FOR f=13 TO 19: PRINT
  INK 3; BRIGHT 1;AT f,31;"Q": N
EXT f: LET l=USR 64958: PRINT  I
NK 3; BRIGHT 1;AT 0,31;"D";AT 20
,31;"A"
 27 RETURN
 28 LET l=USR 65153: PRINT  INK
 3; BRIGHT 1;AT 0,1;"SSSSSSSSSS"
; INK 1; BRIGHT 0;"          ";
INK 3; BRIGHT 1;"SSSSSSSSSS"
 29 RETURN
 30 LET l=USR 65153: PRINT  INK
 3; BRIGHT 1;AT 20,1;"RRRRRRRRRR
"; INK 1; BRIGHT 0;"          ";
INK 3; BRIGHT 1;"RRRRRRRRRR"
 32 RETURN
 33 LET l=USR 65153: FOR f=1 TO
19: PRINT  INK 3; BRIGHT 1;AT f
,0;"P": NEXT f: LET l=USR 64958:
 PRINT  INK 3; BRIGHT 1;AT 0,0;"
C";AT 20,0;"B"
 34 RETURN
 36 LET l=USR 65153: FOR f=1 TO
19: PRINT  INK 3; BRIGHT 1;AT f
,31;"Q": NEXT f: LET l=USR 64958
: PRINT  INK 3; BRIGHT 1;AT 0,31
;"D";AT 20,31;"A"
 37 RETURN
 40 LET l=USR 65153: PRINT  INK
3; BRIGHT 1;AT 0,1;"SSSSSSSSSSSSSSS
SSSSSSSSSSSSSSS"
 42 RETURN
 43 LET l=USR 65153: PRINT  INK
3; BRIGHT 1;AT 20,1;"RRRRRRRRRRRRRRR
RRRRRRRRRRRRRRR"
 45 RETURN
 50 REM 1
 60 CLS : GO SUB 6550
 63 IF llave2=1 THEN  RETURN
 65 LET l=USR 64958: PRINT  INK
6; BRIGHT 1; PAPER 0; FLASH 1;A
T 1,15;"L"
 70 RETURN
100 REM 2
110 CLS : GO SUB 6100
120 RETURN
150 REM 3
160 CLS : GO SUB 6400
170 RETURN
200 REM 4
210 CLS : GO SUB 6100
220 RETURN
250 REM 5
260 CLS : GO SUB 6100
270 RETURN
300 REM 6
310 CLS : GO SUB 6100
320 RETURN
350 REM 7
360 CLS : GO SUB 6100
370 RETURN
400 REM 8
410 CLS : GO SUB 6400
420 RETURN
450 REM 9
460 CLS : GO SUB 6400
470 RETURN
500 REM 10
510 CLS : GO SUB 6050
520 RETURN
550 REM 11
560 CLS : GO SUB 6150
570 RETURN
600 REM 12
610 CLS : GO SUB 6200
620 RETURN
650 REM 13
660 CLS : GO SUB 6550
670 RETURN
700 REM 14
710 CLS : GO SUB 6000
```

```
 720 RETURN
 750 REM 15
 760 CLS : GO SUB 6000
 770 RETURN
 800 REM 16
 810 CLS : GO SUB 6350
 820 RETURN
 850 REM 17
 860 CLS : GO SUB 6200
 870 RETURN
 900 REM 18
 910 CLS : GO SUB 6700
 913 IF llave4=1 THEN  RETURN
 915 LET l=USR 64958: PRINT  INK
 6; PAPER 0; BRIGHT 1; FLASH 1;A
T 1,15;"L"
 920 RETURN
 950 REM 19
 960 CLS : GO SUB 6400
 970 RETURN
1000 REM 20
1010 CLS : GO SUB 6300
1020 RETURN
1050 REM 21
1060 CLS : GO SUB 6150
1070 RETURN
1100 REM 22
1110 CLS : GO SUB 6350
1120 RETURN
1150 REM 23
1160 CLS : GO SUB 6350
1170 RETURN
1200 REM 24
1210 CLS : GO SUB 6350
1220 RETURN
1250 REM 25
1260 CLS : GO SUB 6350
1270 RETURN
1300 REM 26
1310 CLS : GO SUB 6100
1320 RETURN
1350 REM 27
1360 CLS : GO SUB 6000
1370 RETURN
1400 REM 28
1410 CLS : GO SUB 6100
1420 RETURN
1450 REM 29
1460 CLS : GO SUB 6100
1470 RETURN
1500 REM 30
1510 CLS : GO SUB 6050
1520 RETURN
1550 REM 31
1560 CLS : GO SUB 6500
1570 RETURN
1600 REM 32
1610 CLS : GO SUB 6550
1620 RETURN
1650 REM 33
1660 CLS : GO SUB 6400
1670 RETURN
1700 REM 34
1710 CLS : GO SUB 6400
1720 RETURN
1750 REM 35
1760 CLS : GO SUB 6050
1770 RETURN
1800 REM 36
1810 CLS : GO SUB 6150
1820 RETURN
1850 REM 37
1860 CLS : GO SUB 6000
1870 RETURN
1900 REM 38
1910 CLS : GO SUB 6000
1920 RETURN
1950 REM 39
1960 CLS : GO SUB 6000
1970 RETURN
2000 REM 40
2010 CLS : GO SUB 6300
2020 RETURN
2050 REM 41
2060 CLS : GO SUB 6500
2070 RETURN
2100 REM 42
2110 CLS : GO SUB 6450
2120 RETURN
2150 REM 43
2160 CLS : GO SUB 6100
2170 RETURN
2200 REM 44
2210 CLS : GO SUB 6100
2220 RETURN
2250 REM 45
2260 CLS : GO SUB 6000
2270 RETURN
2300 REM 46
2310 CLS : GO SUB 6000
2320 RETURN
2350 REM 47
2360 CLS : GO SUB 6000
2370 RETURN
2400 REM 48
2410 CLS : GO SUB 6000
2420 RETURN
2450 REM 49
2460 CLS : GO SUB 6200
2470 RETURN
2500 REM 50
2510 CLS : GO SUB 6250
2520 RETURN
2550 REM 51
2560 CLS : GO SUB 6150
2570 RETURN
2600 REM 52
2610 CLS : GO SUB 6100
2620 RETURN
2650 REM 53
2660 CLS : GO SUB 6200
2670 RETURN
2700 REM 54
2710 CLS : GO SUB 6150
2720 RETURN
2750 REM 55
2760 CLS : GO SUB 6350
2770 RETURN
2800 REM 56
2810 CLS : GO SUB 6200
2820 RETURN
2850 REM 57
2860 CLS : GO SUB 6150
2870 RETURN
2900 REM 58
2910 CLS : GO SUB 6000
```

```
2920 RETURN
2950 REM 59
2960 CLS : GO SUB 6000
2970 RETURN
3000 REM 60
3010 CLS : GO SUB 6200
3020 RETURN
3050 REM 61
3060 CLS : GO SUB 6450
3070 RETURN
3100 REM 62
3110 CLS : GO SUB 6000
3120 RETURN
3150 REM 63
3160 CLS : GO SUB 6000
3170 RETURN
3200 REM 64
3210 CLS : GO SUB 6350
3220 RETURN
3250 REM 65
3260 CLS : GO SUB 6400
3270 RETURN
3300 REM 66
3310 CLS : GO SUB 6350
3320 RETURN
3350 REM 67
3360 CLS : GO SUB 6350
3370 RETURN
3400 REM 68
3410 CLS : GO SUB 6350
3420 RETURN
3450 REM 69
3460 CLS : GO SUB 6350
3470 RETURN
3500 REM 70
3510 CLS : GO SUB 6300
3520 RETURN
3550 REM 71
3560 CLS : GO SUB 6250
3565 IF llave1=1 THEN  RETURN
3566 LET l=USR 64958: PRINT  INK
 6; PAPER 0; FLASH 1; BRIGHT 1;A
T 1,15;"L"
3570 RETURN
3600 REM 72
3610 CLS : GO SUB 6500
3620 RETURN
3650 REM 73
3660 CLS : GO SUB 6150
3670 RETURN
3700 REM 74
3710 CLS : GO SUB 6400
3720 RETURN
3750 REM 75
3760 CLS : GO SUB 6400
3770 RETURN
3800 REM 76
3810 CLS : GO SUB 6400
3820 RETURN
3850 REM 77
3860 CLS : GO SUB 6400
3870 RETURN
3900 REM 78
3910 CLS : GO SUB 6400
3920 RETURN
3950 REM 79
3960 CLS : GO SUB 6400
3970 RETURN
4000 REM 80
4010 CLS : GO SUB 6050
4020 RETURN
4050 REM 81
4060 CLS : GO SUB 6500
4070 RETURN
4100 REM 82
4110 CLS : GO SUB 6500
4120 RETURN
4150 REM 83
4160 CLS : GO SUB 6500
4170 RETURN
4200 REM 84
4210 CLS : GO SUB 6550
4220 RETURN
4250 REM 85
4260 CLS : GO SUB 6400
4270 RETURN
4300 REM 86
4310 CLS : GO SUB 6400
4320 RETURN
4350 REM 87
4360 CLS : GO SUB 6400
4370 RETURN
4400 REM 88
4410 CLS : GO SUB 6400
4420 RETURN
4450 REM 89
4460 CLS : GO SUB 6400
4470 RETURN
4500 REM 90
4510 CLS : GO SUB 6300
4520 RETURN
4550 REM 91
4560 CLS : GO SUB 6450
4570 RETURN
4600 REM 92
4610 CLS : GO SUB 6300
4620 RETURN
4650 REM 93
4660 CLS : GO SUB 6600
4670 RETURN
4700 REM 94
4710 CLS : GO SUB 6450
4720 RETURN
4750 REM 95
4760 CLS : GO SUB 6400
4770 RETURN
4800 REM 96
4810 CLS : GO SUB 6400
4820 RETURN
4850 REM 97
4860 CLS : GO SUB 6400
4870 RETURN
4900 REM 98
4910 CLS : GO SUB 6400
4920 RETURN
4950 REM 99
4960 CLS : GO SUB 6400
4970 RETURN
5000 REM 100
5010 CLS : GO SUB 6650
5013 IF llave3=1 THEN  RETURN
5015 LET l=USR 64958: PRINT  INK
 6; BRIGHT 1; FLASH 1; PAPER 0;A
T 1,15;"L"
5020 RETURN
5100 REM MUERTE DEL ARQUERO
```

```
5110 LET l=USR 64958: PRINT AT x
,y;"S": PRINT AT x+1,y;"QR": FOR
 f=1 TO 20: NEXT f: PRINT AT x,y
;"Q";AT x+1,y;"S": FOR f=1 TO 10
0: NEXT f
5120 LET vidas=vidas-1
5130 IF vidas=0 THEN  STOP
5140 LET x=10: LET b1=23: LET y=
10: LET q=0: LET e=0: CLS : GO T
O 7040
5190 RETURN
5950 PRINT AT 21,0;"VIDAS:";vida
s;"   PUNTOS:";puntos;"    LLAVES
:";llaves
5970 RETURN
6000 REM DECORADO 1
6010 BORDER 0: PAPER 0: INK 4: B
RIGHT 0: CLS
6020 GO SUB 20: GO SUB 25: GO SU
B 30: GO SUB 28
6040 RETURN
6050 REM DECORADO 2
6060 BORDER 0: PAPER 0: INK 4: B
RIGHT 0: CLS
6070 GO SUB 30: GO SUB 36: GO SU
B 40: GO SUB 20: LET L=USR 64763
: PRINT  INK 7; BRIGHT 1;AT 10,2
5;"GCI";AT 9,25;"FDH";AT 8,26;"E
"
6080 RETURN
6100 REM DECORADO 3
6110 BORDER 0: PAPER 0: INK 4: B
RIGHT 0: CLS
6120 GO SUB 40: GO SUB 20: GO SU
B 25: GO SUB 30
6130 RETURN
6150 REM DECORADO 4
6160 BORDER 0: PAPER 0: INK 4: B
RIGHT 0: CLS
6170 GO SUB 33: GO SUB 28: GO SU
B 30: GO SUB 25
6180 RETURN
6200 REM DECORADO 5
6210 BORDER 0: PAPER 0: INK 4: B
RIGHT 0: CLS
6220 GO SUB 28: GO SUB 30: GO SU
B 36: GO SUB 20: LET L=USR 64763
: PRINT  INK 7; BRIGHT 1;AT 10,2
5;"GCI";AT 9,25;"FDH";AT 8,26;"E
"
6230 RETURN
6250 REM DECORADO 6
6260 BORDER 0: PAPER 0: INK 4: B
RIGHT 0: CLS
6270 GO SUB 30: GO SUB 40: GO SU
B 33: GO SUB 36: LET L=USR 64763
: PRINT  INK 7; BRIGHT 1;AT 10,2
5;"GCI";AT 9,25;"FDH";AT 8,26;"E
"
6280 RETURN
6300 REM DECORADO 7
6310 BORDER 0: PAPER 0: INK 4: B
RIGHT 0: CLS
6320 GO SUB 28: GO SUB 43: GO SU
B 20: GO SUB 36: LET L=USR 64763
: PRINT  INK 7; BRIGHT 1;AT 10,2
5;"GCI";AT 9,25;"FDH";AT 8,26;"E
"
6330 RETURN
6350 REM DECORADO 8
6360 BORDER 0: PAPER 0: INK 4: B
RIGHT 0: CLS
6370 GO SUB 43: GO SUB 28: GO SU
B 20: GO SUB 25: LET l=USR 64763
6380 RETURN
6400 REM DECORADO 9
6410 BORDER 0: PAPER 0: INK 4: B
RIGHT 0: CLS
6420 GO SUB 40: GO SUB 43: GO SU
B 20: GO SUB 25
6430 RETURN
6450 REM DECORADO 10
6460 BORDER 0: PAPER 0: INK 4: B
RIGHT 0: CLS
6470 GO SUB 28: GO SUB 33: GO SU
B 43: GO SUB 25
6480 RETURN
6500 REM DECORADO 11
6510 BORDER 0: PAPER 0: INK 4: B
RIGHT 0: CLS
6520 GO SUB 30: GO SUB 28: GO SU
B 33: GO SUB 36: LET l=USR 64763
: PRINT  INK 7; BRIGHT 1;AT 10,2
5;"GCI";AT 9,25;"FDH";AT 8,26;"E
"
6530 RETURN
6550 REM DECORADO 12
6560 BORDER 0: PAPER 0: INK 4: B
RIGHT 0: CLS
6570 GO SUB 40: GO SUB 30: GO SU
B 33: GO SUB 25
6580 RETURN
6600 REM DECORADO 13
6610 BORDER 0: PAPER 0: BRIGHT 0
: INK 4: CLS
6620 GO SUB 28: GO SUB 33: GO SU
B 36: GO SUB 43: LET l=USR 64763
: PRINT  INK 7; BRIGHT 1;AT 10,2
5;"GCI";AT 9,25;"FDH";AT 8,26;"E
"
6630 RETURN
6650 REM DECORADO 14
6660 BORDER 0: PAPER 0: INK 4: B
RIGHT 0: CLS
6670 GO SUB 40: GO SUB 43: GO SU
B 20: GO SUB 36: LET l=USR 64763
: PRINT  INK 7; BRIGHT 1;AT 10,2
5;"GCI";AT 9,25;"FDH";AT 8,26;"E
"
6680 RETURN
6700 REM DECORADO 15
6710 BORDER 0: PAPER 0: INK 4: B
RIGHT 0: CLS
6720 GO SUB 33: GO SUB 43: GO SU
B 40: GO SUB 25
6730 RETURN
7000 REM BUCLE PRINCIPAL
7010 CLS
7020 LET fin=0: LET q=0: LET pan
talla=45: LET numero=pantalla*50
: LET a=10+INT (RND*2): LET b=10
+INT (RND*2): LET e=0: LET x=10:
 LET y=15: LET a$="AB"
7030 LET w=0: LET llave1=0: LET
llave2=0: LET llave3=0: LET llav
e4=0: LET puntos=0: LET vidas=3:
```

```
 LET b1=23: LET llaves=0
7040 LET numero=pantalla*50: GO
SUB numero
7045 GO SUB 5950
7050 LET k$=INKEY$: LET l=USR 65
153
7060 POKE 23658,8
7070 PRINT  INK 4; BRIGHT 1; PAP
ER 0;AT x,y;a$(1);AT x+1,y;a$(2)
7080 IF k$="Q" THEN  LET y=y-1:
PRINT AT x,y; "G" : LET a$="GH":
PRINT AT x+1,y; "H" : IF y<1 THEN
  LET y=1
7090 IF k$="W" THEN  LET y=y+1:
PRINT AT x,y-1; "A";AT x+1,y-1;
"B": LET a$="AB": IF y>30 THEN
LET y=30
7100 IF k$="O" THEN  LET x=x-1:
PRINT AT x+2,y;" ";AT X+1,Y;"F";
AT X,Y;"E": LET a$="FE": IF x<1
THEN  LET x=1
7110 IF k$="K" THEN  LET x=x+1:
PRINT AT x-1,y;" ";AT X,Y;"C";AT
 X+1,Y;"D": LET a$="CD": IF x>18
 THEN  LET x=18
7120 LET rnd=INT (RND*100): IF r
nd>95 AND ATTR (9,25)=71 THEN  L
ET q=1: GO SUB 8600
7125 IF rnd=37 AND x=7 AND y=18
THEN  GO SUB 9900
7130 IF q=1 THEN  LET puntos=pun
tos+1: GO SUB 8600
7140 IF x=1 OR x=18 OR y=1 OR y=
30 THEN  GO SUB 7500
7220 IF ATTR (x,y)=71 OR ATTR (x
+1,y)=71 THEN  GO SUB 5100
7230 IF llaves=4 AND pantalla=50
 AND w=0 THEN  GO SUB 8700: LET
llaves=0: LET fin=1: LET w=1
7240 IF fin=1 AND x>7 AND x<13 A
ND y=30 AND INKEY$="W" THEN  GO
SUB 8700: GO TO 9000
7250 IF pantalla=60 AND fin=1 TH
EN  LET w=0: LET llaves=4
7490 GO TO 7050
7500 IF ATTR (x+2,y)=1 AND INKEY
$="K" THEN  CLS : LET pantalla=p
antalla+10: LET b1=23: LET q=0:
LET x=2: LET e=0: GO TO 7040
7510 IF ATTR (x-1,y)=1 AND INKEY
$="O" THEN  CLS : LET b1=23: LET
 pantalla=pantalla-10: LET q=0:
LET x=18: LET e=0: GO TO 7040
7520 IF ATTR (x,y-1)=1 AND INKEY
$="Q" THEN  CLS : LET b1=23: LET
 pantalla=pantalla-1: LET q=0: L
ET y=30: LET e=0: GO TO 7040
7530 IF ATTR (x,y+1)=1 AND INKEY
$="W" THEN  CLS : LET pantalla=p
antalla+1: LET b1=23: LET q=0: L
ET y=1: LET e=0: GO TO 7040
7540 IF x>7 AND x<13 AND y=30 AN
D llaves=0 AND pantalla=50 AND I
NKEY$="W" AND fin=1 THEN  GO TO
9000
7550 IF llaves=4 AND pantalla=50
 THEN  GO SUB 8700: LET llaves=0
: LET fin=1
```

```
7560 IF ATTR (x,y)=198 AND panta
lla=71 THEN  LET llaves=llaves+1
: LET llave1=1: LET puntos=punto
s+100: GO SUB 7045
7570 IF ATTR (x,y)=198 AND panta
lla=1 THEN  LET puntos=puntos+50
: LET llaves=llaves+1: LET llave
2=1: GO SUB 7045
7580 IF ATTR (x,y)=198 AND panta
lla=100 THEN  LET puntos=puntos+
150: LET llaves=llaves+1: LET ll
ave3=1: GO SUB 7045
7585 IF ATTR (x,y)=198 AND panta
lla=18 THEN  LET puntos=puntos+5
0: LET llaves=llaves+1: LET llav
e4=1: GO SUB 7045
7590 RETURN
8600 IF q<>1 THEN  RETURN : REM
FLECHA
8610 LET a1=x
8620 PRINT  INK 7; BRIGHT 1;AT a
1,b1;"-"; INK 4; BRIGHT 0;AT a1,
b1+1;" ";AT a1-1,b1+1;" ";AT a1+
1,b1+1;" "
8630 LET b1=b1-1
8640 IF b1<1 THEN  PRINT AT a1,1
;" ": LET q=0: LET b1=23: RETURN

8690 RETURN
8700 FOR f=13 TO 7 STEP -1: FOR
s=1 TO 20: NEXT s: PRINT  INK 1;
 BRIGHT 1;AT f,31;" ": NEXT f
8710 RETURN
9000 REM FIN
9010 CLS
9020 LET cont=0: LET x=18: LET y
 =1
 9030 LET a$="AB": LET s$="GHGHIGIG
HJIGHIJHIGHJIGHJIHIHIJGJ"
 9040 LET l=USR 64958: PRINT  BRI
 GHT 1; INK 4; PAPER 0;AT 20,0;s$
 : LET l=USR 65153: PRINT  INK 6;
  BRIGHT 1; PAPER 0;AT x,y;a$(1);
 AT x+1,y;a$(2)
 9045 IF fin=1 THEN  GO SUB 9200:
  LET fin=0
 9050 LET k$=INKEY$: POKE 23658,8
 9060 IF k$="Q" THEN  LET y=y-1:
 PRINT AT x,y+1;" ": LET a$="GH":
  PRINT AT x+1,y+1;" ": IF y<1 TH
 EN  LET y=1
 9070 IF k$="W" THEN  LET y=y+1:
 PRINT AT x,y-1;" ";AT x+1,y-1;"
 ": LET a$="AB": IF y>15 THEN  LE
 T y=15: LET s$=s$(2 TO )+s$(1):
 LET cont=cont+1
 9080 IF cont=100 THEN  GO TO 930
 0
 9190 GO TO 9040
 9200 LET l=USR 65153: PRINT  INK
  6; PAPER 0; BRIGHT 1;AT 15,13;"
 "MN": PRINT  INK 5; BRIGHT 1; PAP
 ER 0;AT 14,13;"KL";AT 13,13; "IJ"
 9210 LET w=0: LET l=0: LET m$="
  Te estaba esperando Arquero..."
 9220 PRINT  INK 0;AT 11,w;m$
 9225 FOR f=0 TO 31: PRINT  INK 4
 ;AT 11,f; OVER 1;" ": FOR s=1 TO
```

```
5: NEXT s: NEXT f
9230 FOR s=1 TO 50: NEXT s: IF l
=1 THEN  GO SUB 9240: RETURN
9235 FOR f=0 TO 31: PRINT AT 11,
f;" ": NEXT f: LET m$=" Sigue y
llegaras a tu destino": LET l=1:
 GO TO 9220
9240 FOR f=0 TO 31: PRINT AT 11,
f;" ": NEXT f: LET l=USR 64958:
PRINT  INK 5; PAPER 0; BRIGHT 1;
AT 13,13;"MN";AT 14,13;"OP": FOR
 f=1 TO 15: NEXT f: PRINT  INK 5
; BRIGHT 1; PAPER 0;AT 13,13;"QR
";AT 14,13;"ST": FOR f=1 TO 15:
NEXT f
9245 PRINT AT 13,13;"  ";AT 14,1
3;"  ";AT 15,13;"  "
9250 RETURN
9310 LET l=USR 65153: FOR f=15 T
O 31: PRINT  BRIGHT 1; INK 6;AT
18,f;"A";AT 19,f;"B": PRINT AT 1
8,f-1;" ";AT 19,f-1;" ": NEXT f
9320 BORDER 1: PAPER 1: INK 7: B
RIGHT 1: CLS
9330 LET l=USR 64958: PRINT  INK
 7; BRIGHT 1;AT 20,0;"GHIJHGIJGJIHG
IHGHIHIJJHJHJGIGJJI"
9340 PRINT  INK 7; BRIGHT 1; PAP
ER 1;AT 18,0;" EFEFEFEFEFEFEFEFEFEFE
FEFEFEFEF EFEFEFEFEFEFEFEFEFEFEFEFE
FEFEFEF
9350 LET a$="LM": LET l=USR 6476
3: PRINT  INK 3; BRIGHT 1;AT 17,
5;"NO";AT 17,25;"NO": FOR f=13 T
O 16: PRINT  INK 3; BRIGHT 1;AT
f,5;a$;AT f,25;a$: NEXT f: PRINT
  INK 3; BRIGHT 1;AT 12,5;"JK";A
T 12,25; JK"
9360 LET l=USR 64958: FOR f=10 T
O 11: PRINT  INK 6; PAPER 2; BRI
GHT 1;AT f,4;"KKKKKKKKKKKKKKK"; I
NK 7; BRIGHT 0; PAPER 1;"    "; B
RIGHT 1; PAPER 2; INK 6;"KKKKKKK
KK": NEXT f
9370 LET l=USR 64763: PRINT  INK
 4; BRIGHT 1;AT 0,0; "QSTUTTUQSTUQUT
QSUTUTQTSTSTS" ";AT 1,0;"R";AT
1,7;"R";AT 1,11;"R";AT 1,14;"R";
AT 1,20;"R"
9375 LET l=USR 64763: PRINT  INK
 7; BRIGHT 1;AT 9,9;"GC";"I";AT
8,9;"FDH";AT 7,10;"E";AT 9,12;"B
";AT 8,12; "A"
9380 LET l=USR 64763: LET n=14:
FOR f=17 TO 0 STEP -1: PRINT  IN
K 2; BRIGHT 1;AT f,n;"P": LET n=
n+1: NEXT f
9390 LET l=USR 64958: PRINT AT 4
,20;"EF": LET l=USR 65153: PRINT
 AT 3,20;"KL";AT 2,20;"IJ"
9400 PRINT  INK 6; BRIGHT 1;AT 1
8,0;"A";AT 19,0;"B": FOR f=1 TO
10: NEXT f: PRINT AT 19,0;" ";AT
 18,0;" ": PRINT  INK 6; BRIGHT
1;AT 17,1;"A";AT 18,1;"B": FOR f
=1 TO 10: NEXT f: PRINT AT 17,1;
" ": PRINT AT 18,1;" "
9410 PRINT  INK 6; BRIGHT 1;AT 1
6,2;"A";AT 17,2;"B": FOR f=1 TO
10: NEXT f: PRINT AT 16,2;" ";AT
 17,2;" ": PRINT  INK 6; BRIGHT
1;AT 16,3;"A";AT 17,3;"B": FOR f
=1 TO 10: NEXT f: PRINT AT 16,3;
" ": PRINT AT 17,3;" ": FOR f=1
TO 10: NEXT f
9420 FOR f=8 TO 13: PRINT  INK 6
; BRIGHT 1;AT 16,f;"A";AT 17,f;"
B": PRINT AT 16,f-1;" ";AT 17,f-
1;" ": FOR s=1 TO 10: NEXT s: NE
XT f
9430 LET l=13: FOR f=16 TO 0 STE
P -1: PRINT  INK 6; BRIGHT 1;AT
f+1,l;"B";AT f,l;"A": LET l=l+1:
 PRINT AT f,l-1;" ": PRINT AT f+
1,l-1;" ": FOR s=1 TO 10: NEXT s
: NEXT f
9440 BORDER 0: PAPER 0: INK 7: B
RIGHT 1: OVER 0: INVERSE 0: CLS
9450 PLOT  OVER 1;150,87: DRAW
OVER 1;0,50: DRAW  OVER 1;40,0:
DRAW  OVER 1;0,-50: DRAW  OVER 1
;-41,0: FOR f=0 TO 15: PLOT  OVE
R 1;153+INT (RND*38),89+INT (RND
*48): NEXT f
9500 LET l=USR 65153: PRINT  INK
 5; BRIGHT 1;AT 5,14;"IJ";AT 6,1
4;"KL"
9510 PRINT AT 3,0;" Atraviesa la
 puerta Arquero..."
9520 LET l=USR 65153: PRINT  INK
 6; BRIGHT 1;AT 10,0;"A";AT 11,0
;"B"
9530 LET l=USR 64763: PRINT  INK
 7; BRIGHT 1;AT 9,28;"GCI";AT 8,
28;"FDH";AT 7,29;"E"
9535 FOR f=1 TO 50: NEXT f: PLOT
 220,111
9540 DRAW -25,10: PLOT 220,111:
DRAW -25,0: PLOT 220,111: DRAW -
25,-10
9560 FOR f=4 TO 11: FOR n=18 TO
23: PRINT AT f,n;" ": NEXT n: NE
XT f
9570 FOR s=1 TO 10: FOR f=0 TO 7
: OUT 254,f: NEXT f: NEXT s: BOR
DER 0
9580 PAPER 0: INK 7: BRIGHT 0: C
LS
9590 LET l=USR 65153: PRINT  INK
 5; BRIGHT 1;AT 10,15;"IJ";AT 11
,15; "KL"
9595 PRINT AT 8,0;"        Venger
ha vencido": FOR f=1 TO 50: NEXT
 f: PRINT AT 8,0;"No pierdas la
esperanza Arquero ": FOR f=1 TO
100: NEXT f: PRINT AT 7,0;" Otra
 oportunidad tendras en el futur
o Arquero,en tu proxima    avent
ura...": FOR f=1 TO 100: NEXT f
9600 REM MENU
9610 BORDER 0: PAPER 0: INK 5: B
RIGHT 1: CLS
9620 PRINT AT 6,10;"S-JUGAR";AT
8,10;"I-INSTRUCCIONES"
9630 LET k$=INKEY$: POKE 23658,8
9640 IF k$="S" THEN  CLS : RUN 5
9650 IF k$="I" THEN  GO SUB 9700
```

```
9660 GO TO 9630
9700 REM INSTRUCCIONES
9710 BORDER 2: PAPER 2: INK 6: B
RIGHT 1: CLS
9720 LET l=USR 65153: PRINT  INK
 1; BRIGHT 1; PAPER 7; FLASH 1;A
T 0,10;"INSTRUCCIONES": PRINT  I
NK 5; BRIGHT 1;AT 2,15;"IJ";AT 3
,15;"KL": PRINT AT 5,0;"  Escuch
a atento Arquero:          Debes
conseguir las 4 llaves  del vien
to del Sur.Una vez con  ellas de
bes ir hacia el Este y  al entra
r en una determinada    pantalla
 veras que todo sube a  tus pies
..."
9730 PRINT #0;"    Buena suerte..
.Arquero"
9740 PAUSE 0: GO TO 9600
9900 REM AMO DEL CALABOZO
9903 IF llaves<4 THEN  GO TO 991
0
9905 IF llaves=4 THEN  GO TO 995
0
9910 LET l=USR 65153: PRINT AT 1
0,15;"KL";AT 9,15;"IJ": PRINT  I
NK 0; BRIGHT 0;AT 8,1;"    Soy el
 amo del calabozo": FOR f=1 TO 3
0: PRINT  INK 7; BRIGHT 1;AT 8,f
; OVER 1;" ": NEXT f: FOR f=1 TO
 30: NEXT f: FOR f=1 TO 30: PRIN
T AT 8,f;" ": NEXT f
9915 LET q$=" llaves": IF llaves
=3 THEN  LET q$=" llave"
9920 PRINT  INK 0; BRIGHT 0;AT 8
,1;"     Debes encontrar ";4-llav
es;q$: FOR f=1 TO 30: PRINT  INK
 7; BRIGHT 1;AT 8,f; OVER 1;" ":
 NEXT f: FOR f=1 TO 30: NEXT f:
FOR f=1 TO 30: PRINT AT 8,f;" ":
 NEXT f
9930 LET l=USR 64958: PRINT  INK
 6; BRIGHT 1;AT 9,15;"MN";AT 10,
15;"OP": FOR F=1 TO 15: NEXT F:
 PRINT  INK 5; BRIGHT 1;AT 9,15;"
QR";AT 10,15;"S": FOR f=1 TO 15:
 NEXT f: PRINT AT 9,15;"  ";AT 1
0,15;"  "
9940 RETURN
9950 LET l=USR 65153: PRINT AT 1
0,15;"KL";AT 9,15;"IJ": PRINT  I
NK 0; BRIGHT 0;AT 8,1;"    Soy el
 amo del calabozo": FOR f=1 TO 3
0: PRINT  INK 7; BRIGHT 1;AT 8,f
; OVER 1;" ": NEXT f: FOR f=1 TO
 30: NEXT f: FOR f=1 TO 30: PRIN
T AT 8,f;" ": NEXT f
9960 PRINT  INK 0; BRIGHT 0;AT 8
,1;"   Busca la Pantalla del Este
": FOR f=1 TO 30: PRINT  INK 7;
BRIGHT 1;AT 8,f; OVER 1;" ": NEX
T f: FOR f=1 TO 30: NEXT f: FOR
f=1 TO 30: PRINT AT 8,f;" ": NEX
T f
9970 LET l=USR 64958: PRINT  INK
 4; BRIGHT 1;AT 9,15;"MN"; INK 4
; BRIGHT 1;AT 10,15;"OP": FOR F=
1 TO 15: NEXT F: PRINT  INK 6; B
RIGHT 1;AT 9,15;"QR";AT 10,15;"S
": FOR f=1 TO 15: NEXT f: PRINT
AT 9,15;"  ";AT 10,15;"  "
9980 RETURN
```

Aunque el programa pueda parecer un poco largo, los resultados una vez metido y ejecutado, son realmente sorprendentes. Este juego cuenta con más de cien pantallas distintas dentro de las cuales estate seguro que te perderás.

Antes de introducir el segundo programa de los dos que componen el juego, introduce el que aparece a continuación:

```
 10 REM ****************************
 20 REM * PROGRAMA PARA INTRODUCIR *
 30 REM * LOS NUMEROS DE LA FIG. 4 *
 40 REM ****************************
 50 REM
 60 CLS
 70 PRINT "INTRODUCE LOS NUMEROS UNO A UNO Y"
 80 PRINT "PULSA 'ENTER' DESPUES DE CADA UNO"
 90 LET N=0
100 FOR I=64568 TO 65535
110    INPUT "NUMERO = ";A
120    POKE I,A
130    LET N=N+A
140 NEXT I
150 IF N<>64588 THEN GOTO 500
160 CLS
170 PRINT "PREPARA LA CINTA PARA GRABAR Y"
180 PRINT
190 PRINT FLASH 1;"PULSA UNA TECLA"
200 PAUSE 0
```

```
210 SAVE "GRAFICOS" CODE 64568,65536-64568
220 CLS
230 PRINT "OPERACION TERMINADA"
240 GOTO 9999
500 CLS
510 PRINT "HA HABIDO ALGUN ERROR EN LA INTRODUCION"
520 PRINT "DE LOS NUMEROS. INTENTALO DE NUEVO"
```

Este programa te va a servir para introducir todos los números que aparecen en la figura 4.

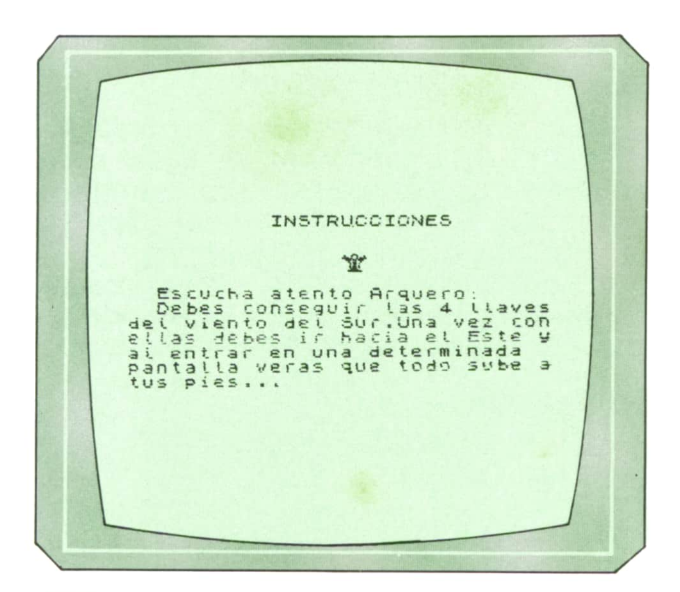
```
1 168 0 33 81 252 17 88 255 237
176 201 0 1 168 0 17 81 252 33 8
8 255 237 176 201 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 168 0 33 20 253 17 88 25
5 237 176 201 0 1 168 0 33 88 25
5 17 20 253 237 176 201 66 90 36
60 24 60 126 189 165 153 60 102
102 102 36 195 189 189 189 60 6
0 60 126 255 36 219 231 231 231
219 189 189 0 0 165 90 90 60 60
36 0 1 1 3 3 3 7 7 7 7 3 3 3 1 1
0 0 128 128 192 192 192 224 224
224 224 192 192 192 128 128 0 0
127 255 127 36 36 36 36 0 248 2
52 248 144 144 144 144 36 36 36
36 36 36 36 36 144 144 144 144 1
44 144 144 144 36 36 36 36 36 36
255 255 144 144 144 144 144 144
252 252 60 74 98 169 147 165 82
60 255 255 255 126 126 62 60 60
60 60 24 24 24 8 8 0 255 255 25
5 126 28 28 8 8 255 253 252 120
56 48 16 0 255 255 143 14 4 4 0
0 0 0 1 168 0 33 215 253 17 88 2
55 237 176 201 0 1 168 0 33 88 2
55 17 215 253 237 176 201 3 3 7
7 15 63 255 255 192 192 224 224
240 252 255 255 255 255 252 240
224 224 192 192 255 255 63 15 7
7 3 3 0 36 91 81 42 36 75 52 0 1
6 236 18 164 74 146 108 0 0 0 4
74 181 72 170 0 0 0 0 5 138 116
137 0 0 4 106 145 66 8 34 0 0 0
0 160 89 166 17 255 64 64 64 255
4 4 4 24 36 24 8 8 56 8 56 0 0
0 0 0 1 16 4 0 0 0 0 128 64 8 16
```

```
0 1 4 0 5 2 0 2 9 32 64 0 160 64
0 160 80 0 0 0 0 0 1 4 0 0 0 0
0 0 0 80 0 0 4 0 0 2 0 0 8 0 32
0 64 0 0 128 16 0 0 0 0 0 0 0 0
0 0 1 168 0 33 153 254 17 88 255
237 176 201 1 168 0 33 88 255 1
7 153 254 237 176 201 0 48 98 66
51 75 107 95 75 50 50 40 36 66
66 99 0 24 164 164 152 164 172 2
46 165 152 164 66 66 66 36 66 0
25 61 61 25 61 189 127 61 25 36
36 36 66 66 66 0 12 70 66 204 21
0 214 250 210 76 76 20 36 66 66
198 0 0 0 0 0 0 1 34 0 0 0 0 0 0
128 68 122 29 9 5 5 5 3 15 94 1
84 144 160 160 160 192 240 0 2 1
3 16 42 40 69 82 0 0 128 240 72
16 72 36 0 0 0 8 4 255 4 8 192 2
24 248 254 254 248 224 192 3 7 3
1 127 127 31 7 3 0 24 24 60 60 1
26 255 255 255 255 126 60 60 24
24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 157 89 76 56 192 87 113 1
4 230 5 205 5 198 30 47 117 118
27 3 19 0 62 0 60 66 66 126 66 6
6 0 0 124 66 124 66 66 124 0 0 6
0 66 64 64 66 60 0 0 120 68 66 6
6 68 120 0 0 126 64 124 64 64 12
6 0 0 126 64 124 64 64 64 0 0 60
66 64 78 66 60 0 0 66 66 126 66
66 66 0 0 62 8 8 8 8 62 0 0 2 2
2 66 66 60 0 0 68 72 112 72 68
66 0 0 64 64 64 64 64 126 0 0 66
102 90 66 66 66 0 0 66 98 82 74
70 66 0 0 60 66 66 66 66 60 0 0
124 66 66 124 64 64 0 0 60 66 6
6 82 74 60 0 0 126 132 132 248 1
36 132 32 0 241 0 240 9 8 240 0
23 249 52 140 20 32 48 38 74 144
144 244 122 1 60 2
```



Estos son los números que tienes que introducir en memoria con ayuda del programa 3.

Estos números que tenemos que introducir son las rutinas en código máquina





Instrucciones del programa «Dragones y mazmorras».

y los caracteres programados por el usuario (UDG) que se utilizan en el programa.

Una vez que estén todos los números metidos en la memoria del ordenador, el programa los grabará en una cinta de cassette.

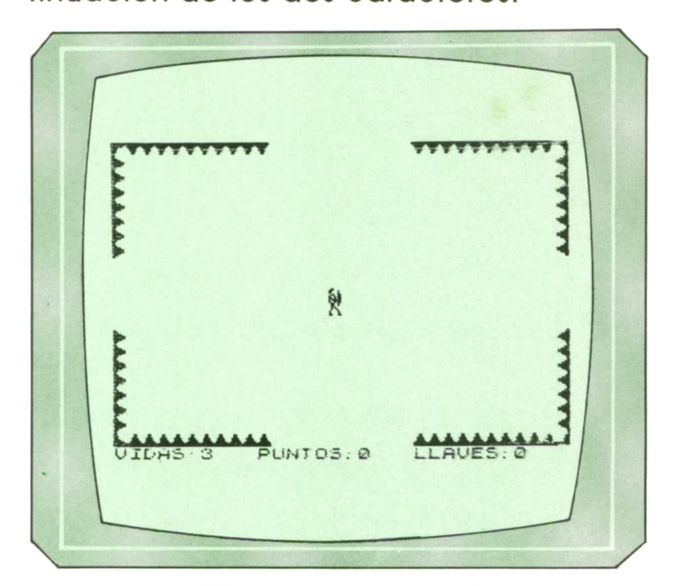
Ahora es el momento de meter el programa más largo. Lo tendrás que grabar a continuación de los números. Según esto, hay que seguir cuatro pasos antes de tener el programa.

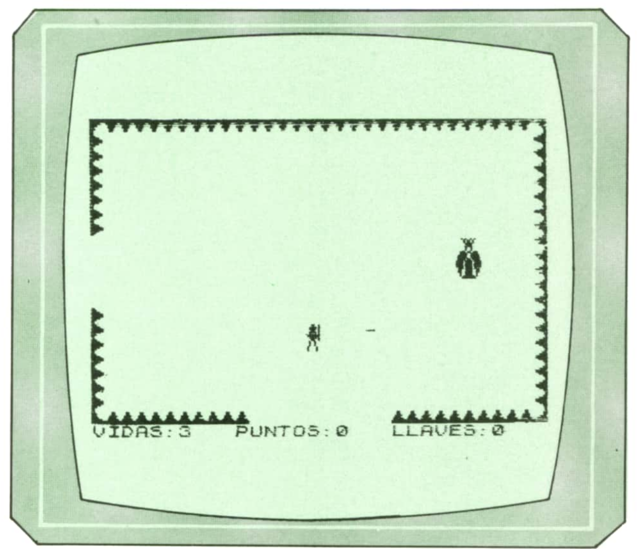
Estos números que tenemos que introducir son las rutinas en código máquina y los caracteres programados por el usuario (UDG) que se utilizan en el programa.

Una vez que estén todos los números metidos en la memoria del ordenador, el programa los grabará en una cinta de cassette.

Ahora es el momento de meter el programa más largo. Lo tendrás que grabar a continuación de los números. Según esto, hay que seguir cuatro pasos antes de tener el programa listo para jugar. Estos son:

1. Introducir el primer programa de los dos que componen el juego y grabarlo.
2. Introducir el programa 3.
3. Ejecutar el programa 3 e introducir los números que aparecen en la figura 4. Después habrá que grabar los números introducidos a continuación de lo que ya habíamos grabado.
4. Introducir el segundo programa que compone el juego y grabarlo a continuación de los dos caracteres.





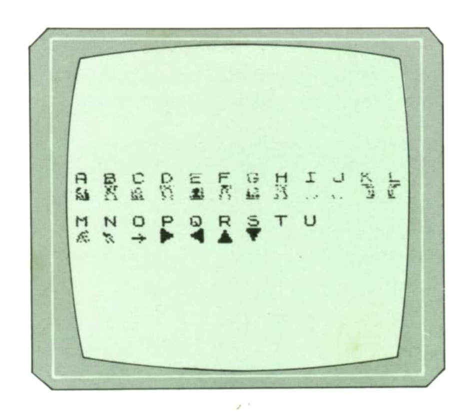


Dos pantallas del juego en plena ejecución.

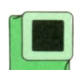
Hay que hacer una advertencia: todas las letras que aparecen subrayadas en el programa no son letras normales del ordenador, sino los caracteres gráficos de éste. Para introducirlos en el programa tenemos que poner el cursor en modo GRAPHICS y pulsar la tecla que corresponda.

Las teclas que nos van a permitir mover a nuestro héroe por la pantalla y de una habitación a otra son:

O Movimiento hacia arriba
K Movimiento hacia abajo
Q Movimiento hacia la izquierda
W Movimiento hacia la derecha




Caracteres definibles utilizados en el programa.


Programa: Estadística

El programa que aparece a continuación está especialmente pensado para aquéllos que, estudiando una carrera o por su trabajo, necesitan hacer cálculos estadísticos con cierta velocidad.

Este puede funcionar perfectamente en el IBM. Para el resto de los ordenadores hay que realizar los siguientes cambios:

COMMODORE:

```
26 PRINT CHR$(147)
59 PRINT CHR$(147)
```

```
1 REM ***************************************************************************
2 REM **********      ESTADISTICA (RECTAS DE REGRESION)          *************
3 REM **********      POR : JUAN MANUEL GUTIERREZ LEITON          *************
4 REM ***************************************************************************
5 REM
6 REM
7 REM ***************************************************************************
8 REM ************* (C) EDICIONES SIGLO CULTURAL (1987)   *******************
9 REM ***************************************************************************
10 REM                este programa sirve para trabajar con variables
11 REM                estadisticas bidimensionales
12 REM x --> x' es la pregunta por si hay cambio en la variable x
13 REM y --> y' es la pregunta por si hay cambio en la variable y
14 REM o(x) es la marca de clase de la variable x que ocupa el lugar central
15 REM c(x) es la amplitud del intervalo de la variable x
16 REM o(y) es la marca de clase de la variable y que ocupa el lugar central
17 REM c(y) es la amplitud del intervalo de la variable y
18 REM a10 es la media de la variable x
19 REM a01 es la media de la variable y
20 REM  m20 es la varianza de la variable x
21 REM  m'20 es la varianza de la variable x cuando ha habido cambio x --> x'
22 REM  m02 es la varianza de la variable y
23 REM  m'02 es la varianza de la variable y cuando ha habido cambio y --> y'
24 REM  m11 es la covarianza
25 DIM Z(1600)
26 CLS
27 INPUT "numero de datos de x ";A
28 INPUT "numero de datos de y ";B
29 INPUT "x -->x' (S/N) ";X$
30 INPUT "y -->y' (S/N) ";Y$
31 IF X$="S" OR X$="s" THEN GOTO 33
32 GOTO 35
33 INPUT "o(x)=";C
34 INPUT "c(x)=";D
35 IF Y$="S" OR Y$="s" THEN GOTO 37
36 GOTO 39
37 INPUT "o(y)=";E
38 INPUT "c(y)=";F
39 FOR G=1 TO A
40   PRINT "x(";G;")=";
41   INPUT Z(G)
42 NEXT G
43 FOR G=1 TO B
44   PRINT "y(";G;")=";
45   INPUT Z(G+A)
46 NEXT G
47 I=0
```

```
48 FOR G=1 TO A
49   FOR H=1 TO B
50      I=I+1
51      PRINT "X";G;"Y";H;
52      INPUT Z(A+B+I)
53    NEXT H
54 NEXT G
55 N=0
56 FOR G=B+A+1 TO B+A+A*B
57    N=N+Z(G)
58 NEXT G
59 CLS
60 LOCATE 12,35
61 PRINT "poblacion=";N
62 GOSUB 225
63 J=0
64 FOR G=1 TO A
65    FOR H=1 TO B
66       J=J+Z(G)*Z(G*A+B+H)
67    NEXT H
68 NEXT G
69 J=J/N
70 L=0
71 FOR G=1 TO B
72    FOR H=1 TO A
73       L=L+Z(A+G)*Z(A+G+B*H)
74    NEXT H
75 NEXT G
76 L=L/N
77 M=0
78 FOR G=1 TO A
79    FOR H=1 TO B
80       M=M+Z(G)*Z(H+A)*Z(H+A+B*G)
81    NEXT H
82 NEXT G
83 M=M/N
84 O=0
85 FOR G=1 TO A
86    FOR H=1 TO B
87       O=O+(Z(G)^2)*Z(G*A+B+H)
88    NEXT H
89 NEXT G
90 O=O/N
91 P=0
92 FOR G=1 TO B
93    FOR H=1 TO A
94       P=P+(Z(A+G))^2*Z(A+G+H*B)
95    NEXT H
96 NEXT G
97 P=P/N
98 IF X$="s" OR X$="S" THEN GOTO 104
99 IF Y$="S" OR Y$="s" THEN GOTO 102
100 GOSUB 223
101 GOTO  106
102 GOSUB 204
103 GOTO 106
104 IF Y$="S" OR Y$="s" THEN GOSUB 157:GOTO 106
105 GOSUB 185
106 CLS
107 LOCATE 8,22
108 PRINT "medias de las variables estadisticas"
109 LOCATE 9,22
110 PRINT "------------------------------------"
111 LOCATE 11,26
112 PRINT "media de la variable x=";J
113 LOCATE 13,26
114 PRINT "media de la variable y=";L
```

```
115 GOSUB 225
116 LOCATE 8,27
117 PRINT "momentos respecto al origen"
118 LOCATE 9,27
119 PRINT "----------------------------"
120 LOCATE 11,35
121 PRINT "a11=";M
122 LOCATE 13,35
123 PRINT "a20=";O
124 LOCATE 15,35
125 PRINT "a02=";P
126 GOSUB 225
127 LOCATE 8,19
128 PRINT "momentos centrales o respecto de la media"
129 LOCATE 9,19
130 PRINT "-------------------------------------------"
131 LOCATE 11,30
132 PRINT"m11=";M-L*J
133 LOCATE 13,30
134 PRINT"m20=";O-J^2
135 LOCATE 15,30
136 PRINT"m02=";P-L^2
137 GOSUB 225
138 LOCATE 8,18
139 PRINT"pendientes de las rectas de regresion"
140 LOCATE 9,18
141 PRINT"--------------------------------------"
142 LOCATE 11,25
143 PRINT"pendiente 1=";(M-L*J)/(O-J^2)
144 LOCATE 13,25
145 PRINT"pendiente 2=";(M-L*J)/(P-L^2)
146 GOSUB 225
147 LOCATE 8,18
148 PRINT "formulas de las rectas de regresion"
149 LOCATE 9,18
150 PRINT "------------------------------------"
151 LOCATE 11,15
152 PRINT "de y sobre x -->  y -";L;"=";(M-L*J)/(O-J^2);" (x -";J;")"
153 LOCATE 13,15
154 PRINT "de x sobre y -->  x -";J;"=";(M-L*J)/(P-L^2);" (y -";L;")"
155 GOSUB 225
156 END
157 CLS
158 LOCATE 4,10
159 PRINT"datos estadisticos cuando hay cambio en las variables x e y"
160 LOCATE 5,10
161 PRINT"------------------------------------------------------------"
162 LOCATE 7,25
163 PRINT "media de x'=";J
164 LOCATE 9,25
165 PRINT "media de y'=";L
166 LOCATE 11,25
167 PRINT "a'11=";M
168 LOCATE 13,25
169 PRINT "a'20=";O
170 LOCATE 15,25
171 PRINT "a'02=";P
172 LOCATE 17,25
173 PRINT "m'11=";M-J*L
174 LOCATE 19,25
175 PRINT "m'20=";O-J^2
176 LOCATE 21,25
177 PRINT "m'02=";P-L^2
178 M=D*F*M+D*E*J+F*C*L+C*E
179 O=(D^2)*O+C^2+2*D*C*J
180 P=(F^2)*P+E^2+2*F*E*L
181 J=D*J+C
```

```
182 L=F*L+E
183 GOSUB 225
184 RETURN
185 LOCATE 4,10
186 PRINT "datos estadisticos cuando hay cambio en la variable x"
187 LOCATE 5,10
188 PRINT "-----------------------------------------------------"
189 LOCATE 7,20
190 PRINT "media de x'=";J
191 LOCATE 9,20
192 PRINT "a'11=";M
193 LOCATE 11,20
194 PRINT "a'20=";O
195 LOCATE 13,20
196 PRINT "m'11=";M-J*L
197 LOCATE 15,20
198 PRINT "m'20=";O-J^2
199 M=D*M+C*L
200 O=O*D^2+C^2+2*C*D*J
201 J=D*J+C
202 GOSUB 225
203 RETURN
204 LOCATE 4,10
205 PRINT "datos estadisticos cuando hay cambio en la variable y"
206 LOCATE 5,10
207 PRINT "-----------------------------------------------------"
208 LOCATE 7,20
209 PRINT "media de y'=";L
210 LOCATE 9,20
211 PRINT "a'11=";M
212 LOCATE 11,20
213 PRINT "a'02=";P
214 LOCATE 13,20
215 PRINT "m'11=";M-J*L
216 LOCATE 15,20
217 PRINT "m'02=";P-L^2
218 P=P*F^2+E^2+2*E*F*L
219 M=M*F+E*J
220 L=L*F+E
221 GOSUB 225
222 RETURN
223 REM esta subrutina no necesita hacer ningun cambio
224 RETURN
225 LOCATE 23,23
226 PRINT "pulse una tecla para continuar"
227 A$=INKEY$
228 IF A$="" THEN GOTO 227
229 CLS
230 RETURN
```

datos estadísticos cuando hay cambio en las variables x e y
media de x' = 14.73661
media de y' = 21.03125
a'11 = 454.2902
a'20 = 515.9777
a'02 = 772.5313
m'11 = 144.3609
m'20 = 298.8101
m'02 = 330.2178
pulse una tecla para continuar


Ejemplo de ejecución del programa de estadística.

```
medias de las variables estadísticas

media de la variable x = 48.20983
media de la variable y = 130.1875

pulse una tecla para continuar
```


Ejemplo de ejecución del programa de estadística.

```
 60 POKE 214,12:POKE 211,1
106 PRINT CHR$(147)
107 POKE 214,8:POKE 211,22
109 POKE 214,9:POKE 211,22
111 POKE 214,11:POKE 211,26
113 POKE 214,13:POKE 211,26
116 POKE 214,8:POKE 211,27
118 POKE 214,9:POKE 211,27
120 POKE 214,11:POKE 211,35
122 POKE 214,13:POKE 211,35
124 POKE 214,15:POKE 211,35
127 POKE 214,8:POKE 211,19
129 POKE 214,9:POKE 211,10
131 POKE 214,11:POKE 211,30
133 POKE 214,13:POKE 211,30
135 POKE 214,15:POKE 211,30
138 POKE 214,8:POKE 211,18
140 POKE 214,9:POKE 211,18
142 POKE 214,11:POKE 211,25
144 POKE 214,13:POKE 211,25
147 POKE 214,8:POKE 211,18
149 POKE 214,9:POKE 211,18
151 POKE 214,11:POKE 211,15
153 POKE 214,13:POKE 211,15
157 PRINT CHR$(147)
158 POKE 214,4:POKE 211,10
160 POKE 214,5:POKE 211,10
162 POKE 214,7:POKE 211,25
164 POKE 214,9:POKE 211,25
166 POKE 214,11:POKE 211,25
168 POKE 214,13:POKE 211,25
170 POKE 214,15:POKE 211,25
172 POKE 214,17:POKE 211,25
174 POKE 214,19:POKE 211,25
176 POKE 214,21:POKE 211,25
185 POKE 214,4:POKE 211,10
187 POKE 214,5:POKE 211,10
189 POKE 214,7:POKE 211,20
191 POKE 214,9:POKE 211,20
193 POKE 214,11:POKE 211,20
195 POKE 214,13:POKE 211,20
197 POKE 214,15:POKE 211,20
204 POKE 214,4:POKE 211,10
206 POKE 214,5:POKE 211,10
208 POKE 214,7:POKE 211,20
210 POKE 214,9:POKE 211,20
212 POKE 214,11:POKE 211,20
214 POKE 214,13:POKE 211,20
216 POKE 214,15:POKE 211,20
225 POKE 214,23:POKE 211,23
229 PRINT CHR$(147)
```

MSX Y AMSTRAD:

Para que este programa pueda funcionar en estos dos ordenadores, lo único que hay que hacer es cambiar de lugar los argumentos de todas las sentencias LOCATE que aparezcan en el programa. Por ejemplo, si tenemos la línea 127:

```
127 LOCATE 8,19
```

los que tengan un MSX o un AMSTRAD tendrán que poner:

```
127 LOCATE 19,8
```

SPECTRUM:

Los usuarios del SPECTRUM sólo tienen que cambiar todas las sentencias LOCATE que aparezcan en el programa por sentencias PRINT AT, conservando el orden de los argumentos y añadiendo un punto y coma (;) al final de la línea. Por ejemplo, si tenemos la línea 127:

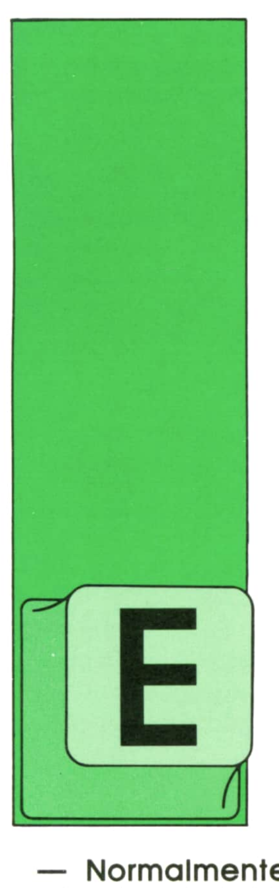
```
127 LOCATE 8,19
```

nosotros tendríamos que cambiarla por:

```
127 PRINT AT 8,19;
```

TECNICAS DE ANALISIS

TABLAS DE DECISION (1)



ES útil señalar que el aspecto concreto de la tabla de decisión suele variar, pero es usual utilizar una serie de normas comunes que conviene tener en cuenta:

— Normalmente, se suelen incluir las reglas tipo AND (Y) las primeras de la tabla; para poner a continuación, en columnas sucesivas, las reglas OR (0) y, por último, la regla ELSE, si hay en la tabla de decisión de que se trate.

— Cuando una condición debe cumplirse en la evaluación de una regla de decisión, se suele poner una S en la fila correspondiente; si *no* debe cumplirse, se suele poner una N (ver figura). Pero puede suceder que esa condición no intervenga (ni sí ni no) en la regla de decisión correspondiente, entonces suele ponerse una raya (—) en la fila, si la regla de decisión es de tipo AND, y un asterisco (*) si la regla es de tipo OR.

— Se llaman funciones de decisión al conjunto de evaluaciones que las definen y por el mismo orden en que se encuentran. Se llaman funciones simples aquellas en que no intervienen los signos — y*; por el contrario, aquellas en que sí intervienen estos signos, se llaman compuestas.

— Es importante comprobar las tablas de decisión para asegurarse de la completud y no redundancia de las informaciones; en efecto, no tiene sentido incluir condiciones que no aparezcan reflejadas con una S, al menos en alguna de las funciones de decisión de la tabla (completud). Del mismo modo, la información no debe ser redundante: las diferentes reglas de decisión deben diferir, al menos, en alguna de las situaciones de decisión. Este proceso de comprobación se complica cuando las tablas son ampliadas y existen funciones de decisión compuestas. Para transformar tablas de decisión en programas, pueden establecerse los siguientes pasos (partiendo de tratamientos elementales):

Situaciones							
Condición 1	S	N	S	S	S		S
Condición 2	N	N	S	S	N		N
Condic. 3 caso A	S	S	S	S	N		S
Condic. 3 caso B	N	N	N	S	N	etc.	S
Condic. 3 caso C	N	N	S	S	S		S
Condición 4	S	S	S	N	N		S
⋮							
Condición m	S	S	N	N	N		N
Tratamiento 1	X	X		X			X
Tratamiento 2		X	X	X	X		
Tratamiento 3 (ir a tabla B)	X			X			X
⋮						etc.	
Tratamiento m		X			X		X



Un ejemplo de tabla de decisión.

a) Escribir todas las situaciones posibles, asignando a cada situación los tratamientos correspondientes y formando

la tabla con reglas simples. El que los tratamientos sean unitarios e independientes es muy importante cara a la simplificación de los procesos y a su posible corrección posterior. Asimismo, es fundamental que las reglas sean unitarias e independientes unas de otras. Esta es, quizá, la fase crítica del diseño de la tabla de decisión y donde ha de ponerse en juego la capacidad de análisis de quien diseña la tabla.

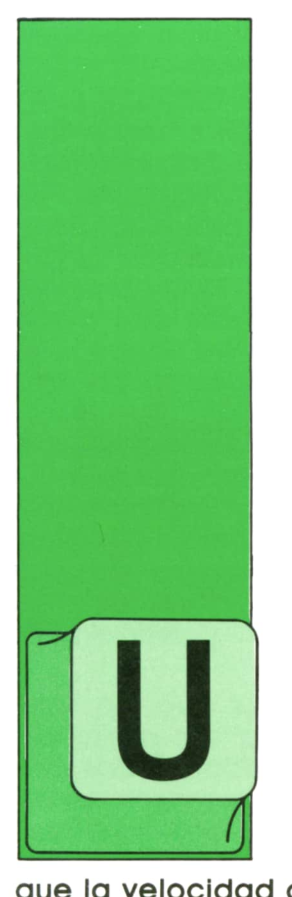
b) Agrupar todas las situaciones que conducen al mismo tratamiento elemental. A veces es muy útil establecer un conjunto flexible de reglas AND y OR articuladas para agrupar el conjunto de situaciones que conducen a un tratamiento; si conceptualmente se refieren a un proceso, deben analizarse conjuntamente, aunque se ubiquen dentro de la tabla en posiciones diferentes. Deben considerarse las diferentes funciones de la tabla de decisión como un esquema lógico de análisis (útil para asegurar la consistencia de los datos, la facilidad de depuración de errores, etc.), pero no como esquema de proceso en el desarrollo del programa: éste debe tener su propia estructura y su propia lógica.

c) Han de simplificarse las reglas simples agrupándolas en reglas compuestas, de acuerdo con el criterio general indicado en el apartado anterior.

d) Pasar a formato tabla, ordenando las reglas de acuerdo con un criterio de frecuencia y número de «indiferencias» que en ella aparecen. (Se considera que aparece en la tabla de decisión una «indiferencia» cuando en la columna correspondiente existe un — o un *; es decir, cuando la condición de que se trate no interviene en la definición de la función de decisión que se está analizando.) Si se conoce la frecuencia de aparición de las reglas de la tabla, se sitúan en las primeras posiciones las reglas para las que el producto de la frecuencia de aparición por el número de indiferencias sea mayor. Si no se conoce la frecuencia de las reglas, se escriben primero las reglas con mayor número de indiferencias.

e) Por último, al pasar al ordinograma correspondiente, se ha de trasladar primero la primera regla y de ésta, la condición que contenga menos indiferencias.

TECNICAS DE PROGRAMACION





Datos escalares

UNA de nuestras actividades más frecuentes en la vida diaria es la toma de datos y la realización de medidas. Hablamos, por ejemplo, de que un objeto pesa 2 kg, de que la velocidad de un automóvil es de 50 km por hora o de que la temperatura ambiente es igual a 27° C. Al realizar todas estas medidas y expresarlas mediante números, estamos utilizando (aunque a menudo no somos conscientes de ello) una escala de medidas apropiada. Por ejemplo, al medir la masa de los objetos utilizamos en el sistema internacional de medidas de masa, basado en el kilogramo, y la medida de 2 kg marca una posición dentro de esta escala. Al mencionar la temperatura hacemos uso de la escala Celsius. Y así sucesivamente.

Sin embargo, un mismo dato podría representarse por dos medidas diferentes dependiendo de la escala que utilicemos para medirlo. Por ejemplo, la misma temperatura ambiente se representa por la cifra de 27° C en la escala Celsius, y por la de 300° Kelvin en la escala de grados absolutos. Los dos números, 27 y 300, se refieren, por tanto, a la misma situación real. Pero para darnos cuenta de ello es preciso mencionar la escala de medida que se ha utilizado para obtener cada dato. En efecto, es evidente que 27 no es igual a 300, considerados como números aislados. Pero también es obvio, para quien conozca las escalas de temperatura, que 27° C es lo mismo que 300° K.

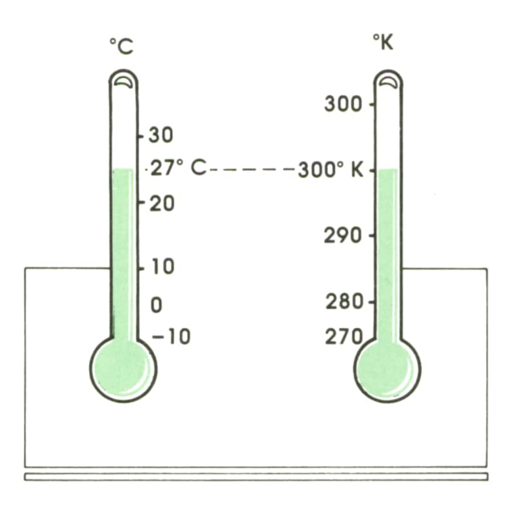




Fig. 1.

Todas estas magnitudes, que pueden representarse mediante un número que depende de la escala de medida que se utilice en cada caso, se llaman «magnitudes escalares». También se dice que el dato concreto es un «escalar». Como puede imaginarse, la palabra «escalar» viene de «escala». En Matemáticas se llama escalar a toda magnitud susceptible de medida, que no tiene dirección.

Vamos a considerar ahora un caso algo más complicado: una longitud de 2 cm ¿es un escalar? Es evidente que se trata de una magnitud susceptible de medida, pero ¿tiene dirección? A primera vista podría parecernos que la longitud siempre tiene dirección. Pero pensemos en esto: supongamos que decimos que una línea trazada en una hoja de pa-

pel tiene una longitud de 2 cm. ¿Hemos dicho algo sobre su dirección? Es evidente que no. La línea podría ir de arriba a abajo, de derecha a izquierda, estar muy cerca de un borde, en el centro del papel, etc., manteniendo siempre la longitud de 2 cm. Por tanto, la medida de longitud no tiene dirección, y se trata también en este caso de un dato escalar.



Vectores

Ahora bien, si marcamos sobre una regla dos puntos separados por una distancia de 2 cm y especificamos la situación del punto de partida y el de llegada, sí estamos dando información sobre la dirección. Si la línea comienza, por ejemplo, en la marca de 12 cm y termina en la de 14 cm, el par (12, 14) nos indica que la longitud de la línea es de 2 cm, pero también especifica algo más, pues nos da la posición exacta que ocupan esos 2 cm sobre la regla. Es decir, tenemos información sobre la dirección. Por otra parte, obsérvese que este par de datos nos da también información sobre el sentido del movimiento al pasar del punto de partida al de llegada. En efecto, el par (12,14) indica que comenzamos en la marca 12 cm y avanzamos hasta la marca 14 cm, mientras que el par (14,12) nos hará recorrer en la misma línea en

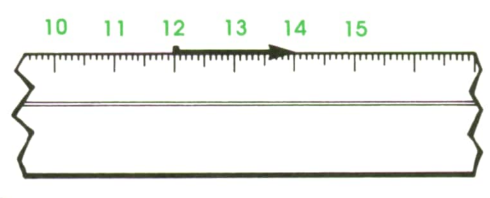



Fig. 2.

sentido contrario. Por todas estas razones, el dato (12,14) no puede considerarse como un escalar, sino como un «vector».

En Matemáticas se llama vector a todo conjunto de datos que define la magnitud, la dirección y el sentido de una medida. La palabra vector viene del latín y significa «el que conduce», pues los datos que constituyen un vector (como en el par de números 12,14) nos «conducen» desde el principio hasta el final de la magnitud provista de dirección y de sentido a la que están midiendo.

Veamos otro ejemplo. Supongamos que trazamos una línea de 2 cm de longitud, pero esta vez la línea no está marcada en el borde de una regla, sino dibujada en una hoja de papel en blanco. ¿Cómo definiremos exactamente su posición, dirección y sentido? Muy sencillo: igual que en el caso anterior, basta con dar la posición del punto de partida y el final de la línea. Pero ¡cuidado! En este caso estos puntos no quedan determinados por un solo número, como en el caso del borde de la regla. Veamos por qué. Supongamos que decimos que el punto de partida se encuentra a 10 cm del vértice inferior izquierdo del papel. Pero ¿en qué dirección debemos movernos esos 10 cm? ¿Horizontalmente? ¿Verticalmente? ¿En diagonal?

No es difícil comprobar que, en este caso, la posición exacta del punto de partida queda determinada por dos medidas de longitud: las distancias de dicho punto a dos de los bordes del papel que sean perpendiculares entre sí. Por ejemplo, podemos decir que el punto de partida de la línea se encuentra a 10 cm del borde izquierdo y a 7 cm del borde inferior. Del mismo modo, el punto final puede estar a 10 cm del borde izquierdo y a 9 cm del inferior. Por tanto, el par (10,7) define la posición del punto de partida y el par (10,9) la del punto final de la línea, y el conjunto de datos ((10,7), (10,9)) define la longitud (2 cm), la dirección (vertical) y el sentido (de abajo a arriba) de la línea entera. Se trata, por consiguiente, de un vector.

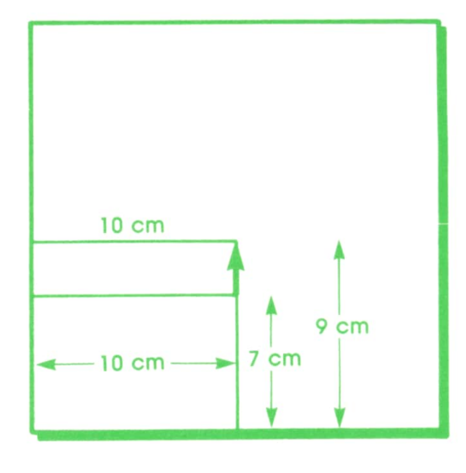



Fig. 3.

De igual manera puede verse que, para definir perfectamente la magnitud, la dirección y el sentido de una línea situada en el espacio, es preciso dar la posición del punto de partida y del punto final, pero ahora cada una de estas posiciones vendrá determinada por tres distancias a tres planos perpendiculares entre sí: los que definen la longitud, la anchura y la profundidad. Por tanto, un vector en el espacio estará formado por un par de datos, cada uno de los cuales consta de tres números diferentes. Por ejemplo: ((2,4,6), (2,4,8)) define una línea de 2 cm de largo situada en dirección de la profundidad y en el sentido de las profundidades crecientes.

Se llama sistema de referencia el conjunto de líneas, puntos o planos que utilizamos como base de partida para medir distancias. En el caso del borde de la regla, medíamos las distancias respecto al punto de la regla marcado con el cero, por lo que nuestro sistema de referencia era dicho punto. En el caso de la hoja de papel, medíamos las distancias respecto al borde inferior y el borde izquierdo de la hoja; luego estas dos rectas constituían nuestro sistema de referencia en este caso. Por último, en el ejemplo del espacio utilizamos como sistema de referencia tres planos perpendiculares entre sí (véase la fig. 4).

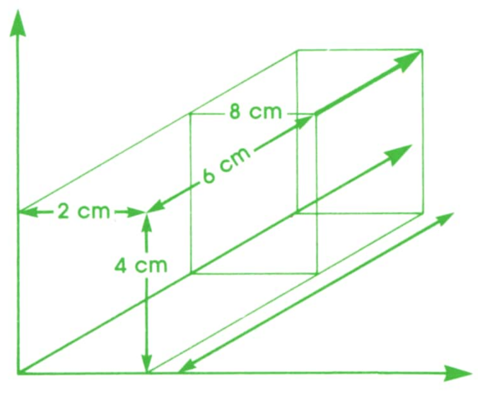




Fig. 4.

En la práctica, siempre puede elegirse el sistema de referencia donde más nos convenga. Por ejemplo, en el caso del segmento situado sobre el borde de la regla, podríamos utilizar como base de las medidas el punto señalado con la marca de 12 cm (el origen de nuestro vector). El segmento quedará ahora definido por un par de números (0,2), pues el primer punto coincide con la base de referencia (su distancia a ella es cero) y el segundo se encuentra a 2 cm de distancia. De igual manera podríamos tomar como sistema de referencia en la hoja de papel dos líneas perpendiculares que pasen por el origen del segmento cuya dirección y magnitud queremos determinar, con lo que el vector quedaría transformado en ((0,0), (0,2)). Finalmente, en el espacio, tomaríamos tres planos perpendiculares que se corten precisamente en el origen del vector, con lo que éste quedaría definido por ((0,0,0), (0,0,2)).

Obsérvese que, de acuerdo con este convenio, el origen del vector queda siempre representado por uno o varios ceros. Por tanto, podemos prescindir de él. El vector quedará perfectamente definido por las distancias desde el punto final al punto inicial (el origen) en las diversas direcciones. Así, pues, el segmento trazado sobre el borde de la regla vendría representado únicamente por el número (2). La línea dibujada en la hoja de papel por el par (0,2). La línea trazada en el espacio por la tríada (0,0,2).

Recapitulando:

1. Sobre una línea recta (como el borde de una regla), para definir totalmente la longitud, la dirección y el sentido de un segmento dado basta con dar un solo número: (2) en nuestro ejemplo.
2. Sobre un plano (como la hoja de papel), para definir totalmente la longitud, la dirección y el sentido de un segmento dado hay que dar dos números: (0,2) en nuestro ejemplo.
3. En el espacio, para definir totalmente la longitud, la dirección y el sentido de un segmento dado hay que dar tres números: (0,0,2) en nuestro ejemplo.

Veamos otro ejemplo. El segmento de la figura 5 quedará definido, en el sistema de referencia que pasa por su origen, por el vector (2,2). De aquí se puede deducir su magnitud (2.828427...), su dirección (una inclinación de 45°) y su sentido (de abajo a arriba).

Vamos a fijarnos ahora en lo siguiente: una línea recta tiene una sola dimensión; un plano tiene dos dimensiones, el espa-

cio tiene tres dimensiones. Por otra parte, un vector sobre la recta queda definido por un número; un vector sobre el plano por dos números; un vector en el espacio por tres números. Esto no es casualidad.

En Matemáticas se utilizan a menudo espacios de más de tres dimensiones. Pues bien: un vector queda siempre definido en estos espacios por tantos números como dimensiones tiene el espacio correspondiente.

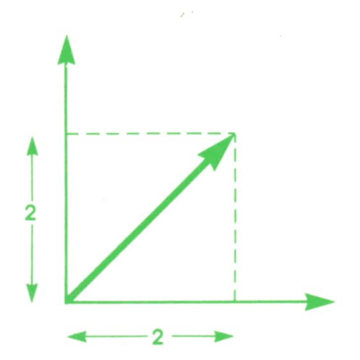




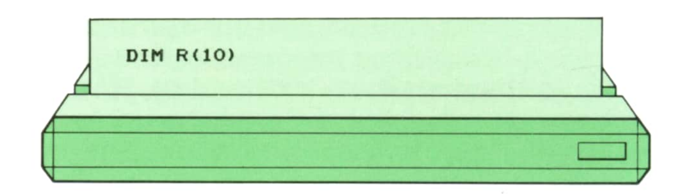
Fig. 5.

Y todo esto, ¿qué tiene que ver con la Informática? Pues bien: es frecuente que los datos que tenemos que utilizar en nuestros programas no sean números aislados, sino colecciones de éstos. Por ejemplo, recordemos la versión BASIC del programa de conversión de números decimales a la base 2:

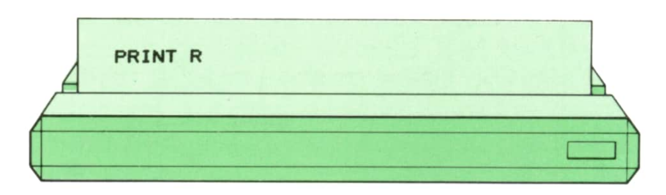
```
10 DIM R(10)
20 FOR I=1 TO 10: LET R(I)=0: NEXT I
30 PRINT "dame un número entre 0 y
   1000"
40 INPUT X
50 FOR I=1 TO 10
60 LET C=INT(X/2)
70 LET R(I)=X-2*C
80 IF C=0 THEN GOTO 110
90 LET X=C
100 NEXT I
110 FOR I=1 TO 10: PRINT R(11-I);:
    NEXT I
```

Podemos ver en este programa que la variable X tiene siempre como valor un número entre 0 y 1.000; la variable C va recibiendo los cocientes sucesivos de X al dividirlo por 2; y la variable I es un contador cuyo valor es un número comprendido entre 1 y 10.

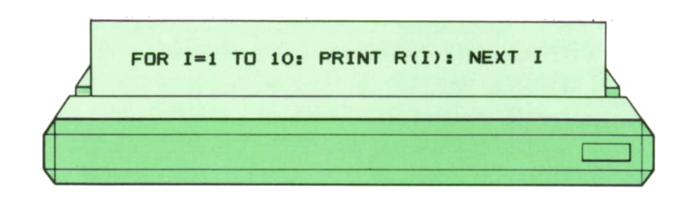
Estas tres variables tienen siempre, por tanto, un valor único. Sin embargo, vemos también que R es una variable diferente. En primer lugar, es preciso definirla mediante una instrucción especial:



En segundo lugar, no tiene un solo valor, sino diez distintos simultáneamente: todos los restos que vamos obteniendo de forma sucesiva al dividir entre 2 el número inicial (X) y los cocientes que lo van sustituyendo. Finalmente (al menos en BASIC), no podemos obtener el valor de R simplemente con la expresión:



como ocurre con las otras variables, que tienen un solo valor. Es preciso utilizar una instrucción más compleja, un bucle:



aunque en la línea 110 del programa anterior, la instrucción era un poco diferente, pues queríamos obtener los valores en orden inverso. En efecto: si I varía de 1 a 10, la expresión (11-I) variará de 10 a 1.

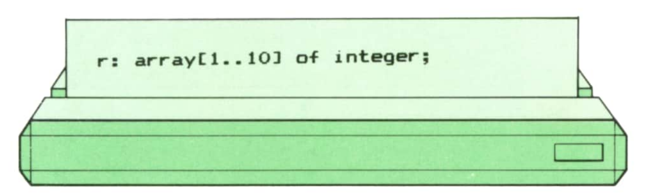
Pues bien: estas variables especiales, que poseen varios valores, se llaman a veces «colecciones», «series» y otros nombres semejantes, pero también es frecuente denominarlas «vectores», precisamente porque tienen la misma estructura que los vectores geométricos. Se dice, además, que el número de valores que poseen es el número de «dimensiones» del vector, puesto que, como vimos antes, un vector tiene siempre tantos elementos como dimensiones tiene el espacio sobre el que define la magnitud, di-

rección y sentido de un segmento determinado.

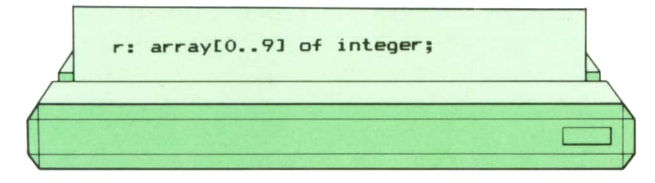
Recordemos ahora la versión PASCAL del mismo programa:

```
program BASE2;
var
  x, i: integer;
  r: array[1..10] of integer;
begin
  writeln('Dame un número entre 0 y
1000'); readln(x);
      for i:=1 to 10 do
        begin
          r[i]:= x mod 2;
          x:=x div 2;
        end;
      for i:=1 to 10 do write(r[11-i]);
end.
```

Podemos ver que también existe una instrucción de declaración de la variable r, que ha de contener los restos, aunque es un poco diferente a la versión BASIC:

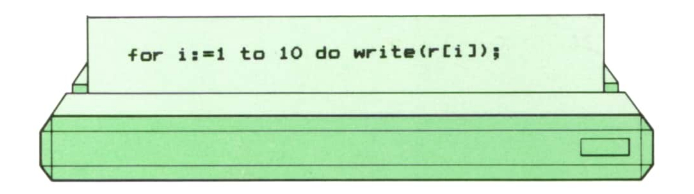


La palabra inglesa «array» se puede traducir como «disposición», «secuencia», etc., y en este caso equivale a «vector». La expresión 1..10 indica que este vector tiene 10 elementos, cuyos índices son los números comprendidos entre 1 y 10. No hay nada en PASCAL que prohíba declaraciones como la siguiente:

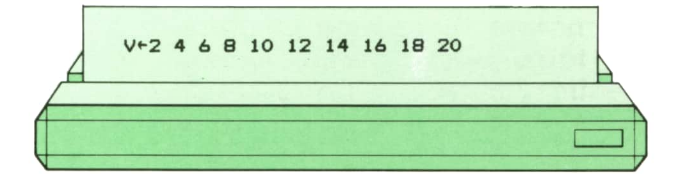


donde r sigue siendo un vector de 10 elementos, pero donde los índices de éstos no son los números del 1 al 10, sino del 0 al 9.

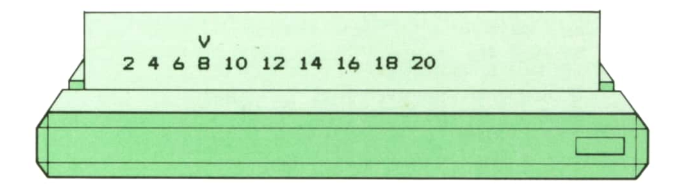
También en el caso de PASCAL tenemos que recurrir a un bucle para visualizar los diez valores de r:



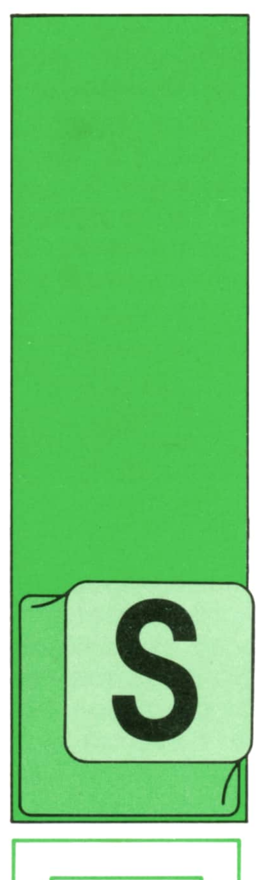
En cambio, en APL no existen instrucciones especiales para declarar vectores ni son necesarios bucles para imprimirlos, pues este lenguaje trabaja de forma natural con estas estructuras de datos. Por ejemplo, si queremos que V sea un vector de diez elementos iguales a los diez primeros números pares, bastaría escribir:

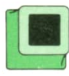


y para visualizar los diez valores de V bastará escribir:



LOGO





Cómo pintar o no pintar

Si queremos realizar el siguiente dibujo:

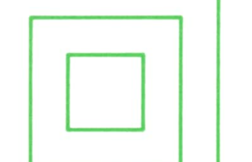




Fig. 1.

no tendríamos ninguna pega para dibujar los diferentes cuadrados, pero nos surgiría un problema. Después de dibujar un cuadrado, para pasar a hacer el siguiente hemos de dejar una parte sin pintar, es decir, hemos de conseguir que la tortuga ande sin dejar rastro.

Para ello, existe un comando que le ordena a la tortuga que levante su lápiz y que no pinte hasta que nosotros se lo digamos. Este comando es:

SL

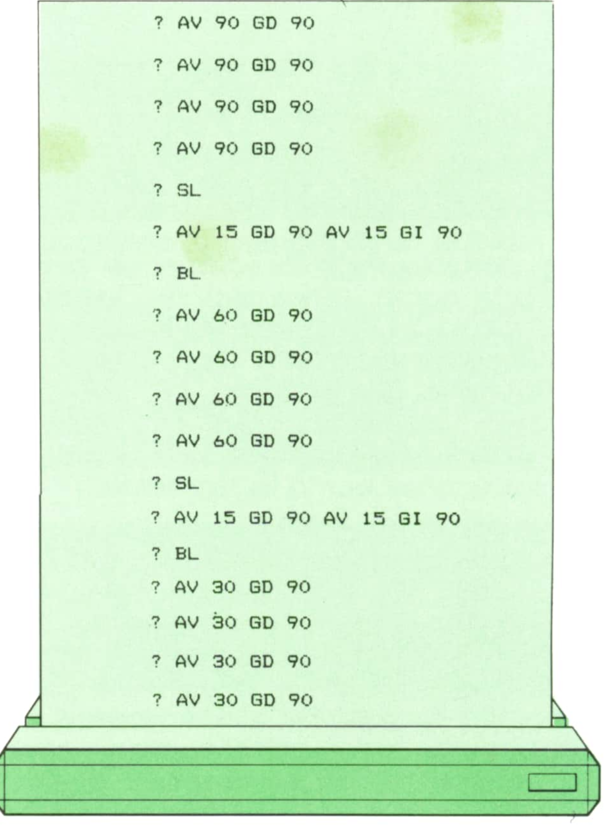
que es la abreviatura de Sube Lápiz.

Cuando ya hayamos colocado a la tortuga en la posición donde queremos que vuelva a pintar, le hemos de decir que baje el lápiz con el comando:

BL

que es la abreviatura de Baja Lápiz.

Ahora ya podemos hacer nuestro dibujo. Las órdenes son:



```
? AV 90 GD 90
? AV 90 GD 90
? AV 90 GD 90
? AV 90 GD 90
? SL
? AV 15 GD 90 AV 15 GI 90
? BL
? AV 60 GD 90
? AV 60 GD 90
? AV 60 GD 90
? AV 60 GD 90
? SL
? AV 15 GD 90 AV 15 GI 90
? BL
? AV 30 GD 90
? AV 30 GD 90
? AV 30 GD 90
? AV 30 GD 90
```



Cómo borrar

Si al realizar alguno de nuestros dibujos nos equivocamos, con los comandos de borrar que conocemos hasta ahora (BP y LIMPIA) no nos queda más remedio que borrar todo el dibujo. Pero es una pena

que por confundirnos en una línea tengamos que empezar de nuevo.

Por ello, la tortuga dispone de una goma o borrador. Cuando le decimos que lo active, en lugar de dejar o no rastro al andar lo que hace es borrar las líneas que existan conforme pasa por encima de ellas.

Para indicarle a la tortuga que borre, se utiliza el comando:

GOMA

Para que la tortuga deje de borrar hemos de decirle SL o BL.

Vamos a probarlo. Supongamos que estamos dibujando un cuadrado:

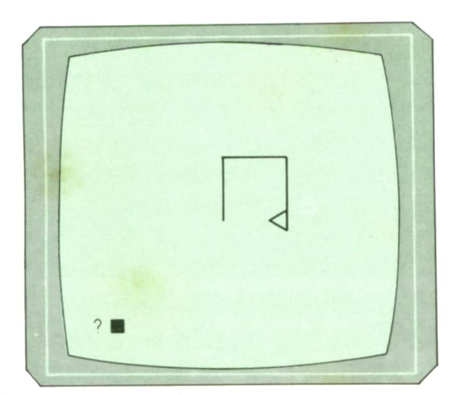
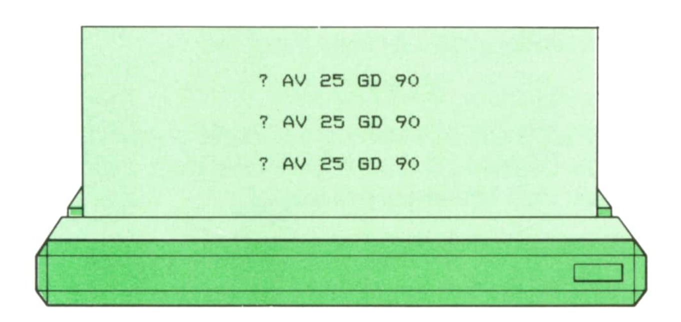


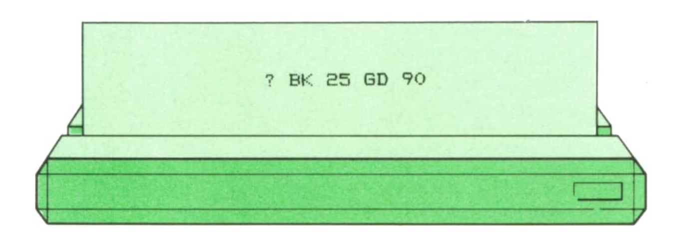


Fig. 2.

Los comandos que habremos escrito son:



Si ahora escribimos:



nos quedará:

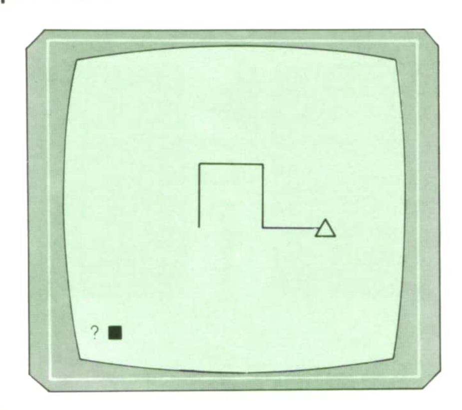
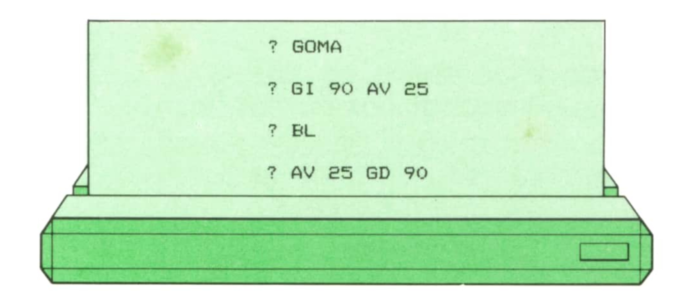


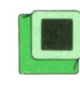


Fig. 3.

y esto no es un cuadrado.

Para no borrar todo el dibujo, le decimos a la tortuga que queremos borrar sólo una parte y hacerlo bien. Las órdenes serían:





La tortuga aparece y desaparece

Cuando hemos finalizado cada uno de los dibujos que hemos hecho, hubiera quedado mejor que sólo se viera el dibujo y que la tortuga no apareciera.

Existe la posibilidad de decirle a la tortuga que se esconda, que se haga invisible hasta que nosotros le digamos lo contrario. Sin embargo, hay que tener en cuenta que aunque nosotros no la veamos, la tortuga sigue ahí y si le decimos que ejecute algún comando lo hará igual que siempre.

El comando que le dice a la tortuga que se oculte es

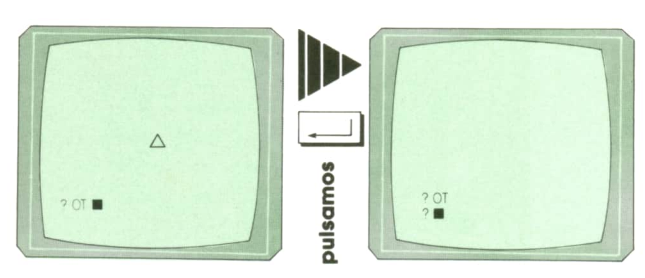
OT

y si deseamos que vuelva a aparecer

MT

Estas palabras son las abreviaturas de Oculta Tortuga y Muestra Tortuga.

Vamos a comprobar que la tortuga obedece estas órdenes:



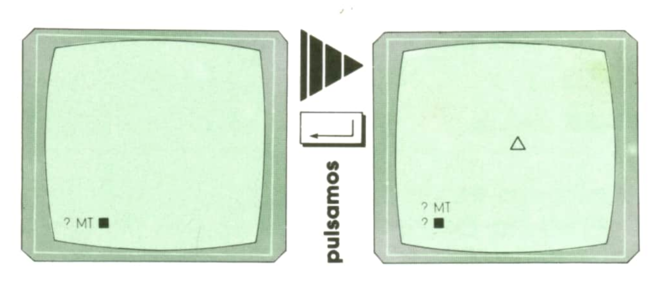
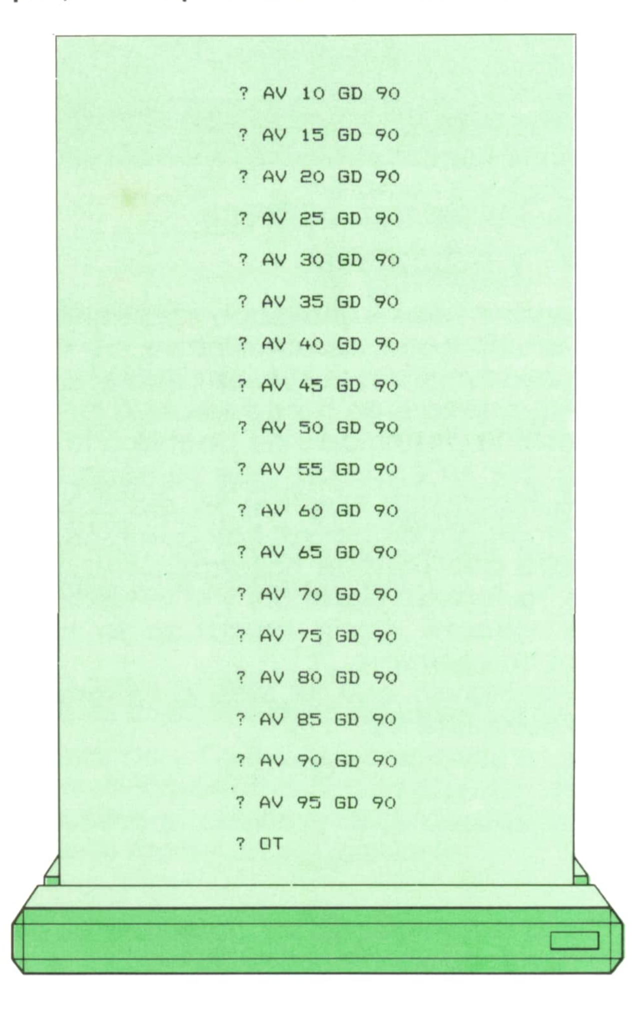



Fig. 4.

Hagamos ahora un dibujo, por ejemplo, una espiral. Los comandos son:



y la figura que nos queda es:

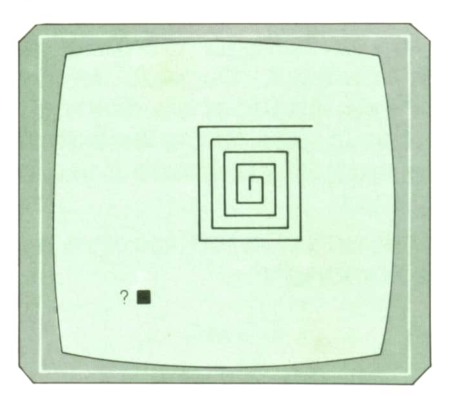



Fig. 5.



Distintos colores

Cuando nosotros escribimos o pintamos, no siempre utilizamos el mismo bolígrafo, sino que tenemos "bolis" y lapiceros de diferentes colores. A la tortuga le ocurre lo mismo: dispone de diferentes tinteros que le permiten que su lápiz pinte en distintos colores.

Para decirle a la tortuga que queremos que cambie el color de su lápiz usamos el comando

PONCL n

abreviatura de PON Color Lápiz y donde n es un número que indica el color en el que la tortuga dejará su rastro a partir de este momento.

Además de poder cambiar el color del lápiz de la tortuga, también podemos variar el color del fondo de la pantalla. El comando para hacer esto es:

PONFONDO n

siendo n un número que indica el color.

Los números correspondientes a cada color son los siguientes:

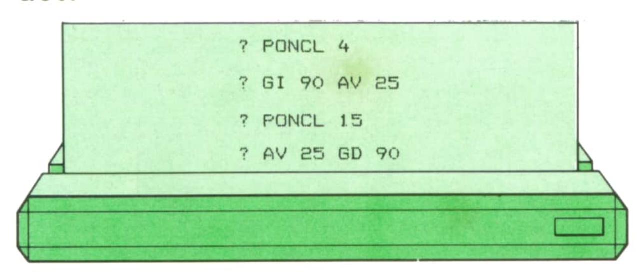
0 Transparente.
1 Negro.
2 Verde.
3 Verde claro.
4 Azul oscuro.
5 Azul claro.
6 Rojo oscuro.
7 Azul celeste.
8 Rojo.
9 Rojo claro.
10 Amarillo oscuro.
11 Amarillo claro.
12 Verde oscuro.

13 Magenta.
14 Gris.
15 Blanco.

En el caso de que el número n que pongamos en los comandos PONCL y PONFONDO sea el mismo, la tortuga dejará rastro al andar, pero nosotros no lo veremos, por ser el color del lápiz el mismo que el color del fondo.

Por tanto, acabamos de descubrir otra forma de borrar líneas cuando nos confundamos. Para ello, nos basta con cambiar el color del lápiz al color que tenga el fondo y hacer pasar a la tortuga por encima de la línea equivocada.

Por ejemplo, en el caso del cuadrado que veíamos antes, obtendríamos el mismo resultado con los siguientes comandos:



Como se ve, hemos supuesto que el color del fondo es el azul oscuro y que la tortuga estaba pintando el cuadrado en color blanco.



Os proponemos

1. Intenta pintar la siguiente figura de tal manera que cada triángulo sea de un color diferente.





Fig. 6.

2. Haz que la tortuga escriba tu nombre en la pantalla. Si quieres, puedes pintar cada letra de un color. Por ejemplo:



Fig. 7.

3. También puedes dibujar este comedor usando distintos colores:

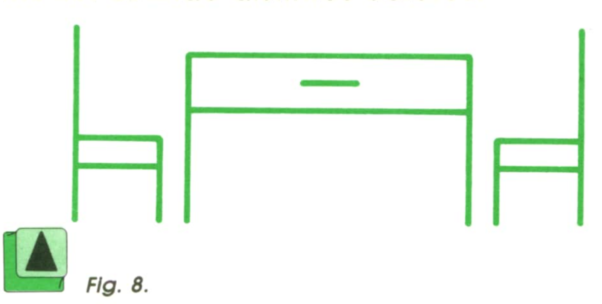


Fig. 8.

4. Pinta este chino:

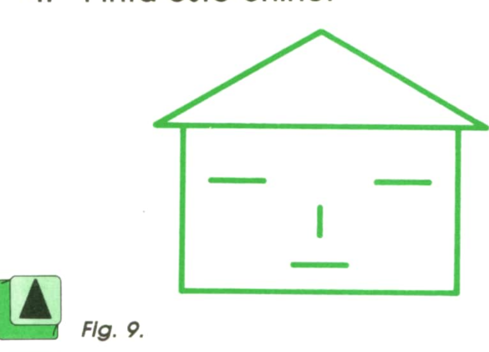


Fig. 9.



Para escribir

Además de dibujar, nuestra tortuga es capaz de hacer muchas más cosas, que iremos viendo poco a poco.

Entre ellas, la tortuga puede escribir lo que nosotros le pidamos. El comando de escritura es:

ESCRIBE

o su abreviatura:

ES

Pero nos falta decirle qué es lo que ha de escribir.

Si deseamos que escriba una palabra, hemos de poner:

ES "palabra

es decir, ponemos dobles comillas (") delante de la palabra.

Por ejemplo, si queremos que escriba HOLA:

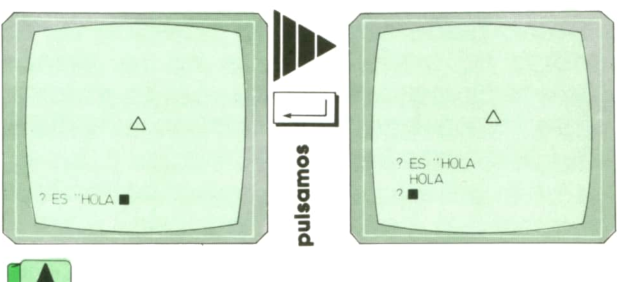




Fig. 10.

Por el contrario, si lo que queremos que escriba es una frase o una lista de palabras, hemos de decirle:

ES (lista o frase)

es decir, encerramos la lista o frase entre corchetes (( )).

Por ejemplo, si queremos que escriba la lista de nuestros amigos, pondremos:

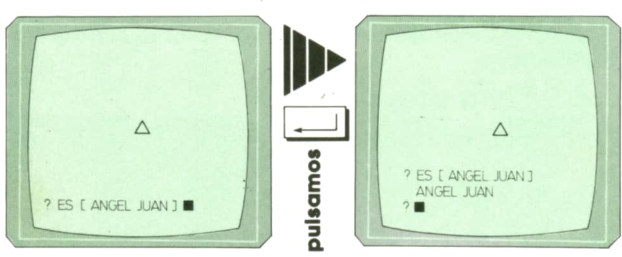




Fig. 11.

o si queremos escribir una frase:

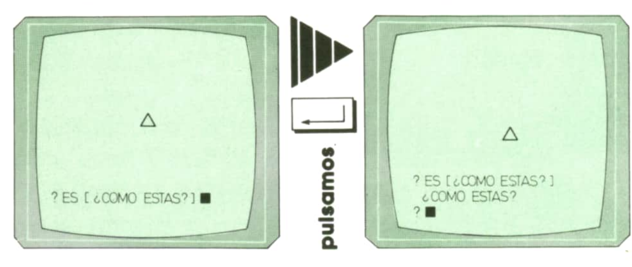




Fig. 12.



Algunas funciones

Una FUNCION es un tipo de orden que la tortuga es capaz de ejecutar pero que no varía su estado, es decir, que no cambia su posición. Más bien, lo que ocurre es que nos sirve para conocer alguna información. Es como si a la tortuga le preguntáramos algo y ella nos respondiese.

Por ejemplo, si deseamos conocer el número del color del lápiz que actualmente está utilizando la tortuga, usaremos la función:

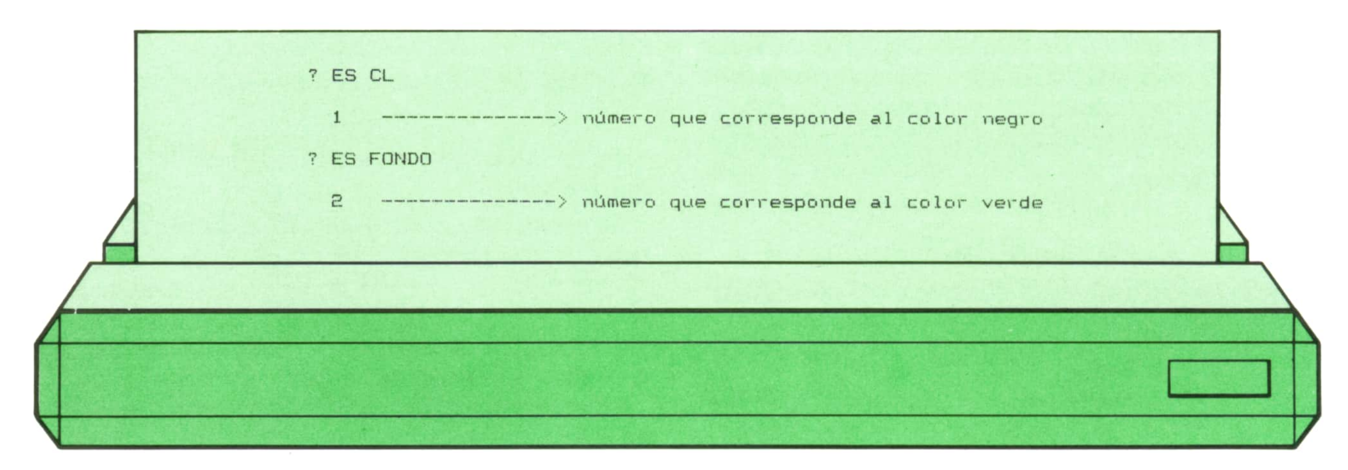
CL

y si queremos conocer el número del color del fondo de la pantalla, la función es:

FONDO

Sólo nos queda tener una cosa en cuenta. Las funciones no se utilizan como los comandos, sino que hay que hacer algo con ellas, con el resultado que dan. Lo más normal es decirle a la tortuga que escriba, mediante el comando ES, la respuesta a lo que nosotros le preguntamos.

Así, si suponemos que el color del fondo es el verde y el color del lápiz de la tortuga el negro, al dar las siguientes órdenes obtendríamos:

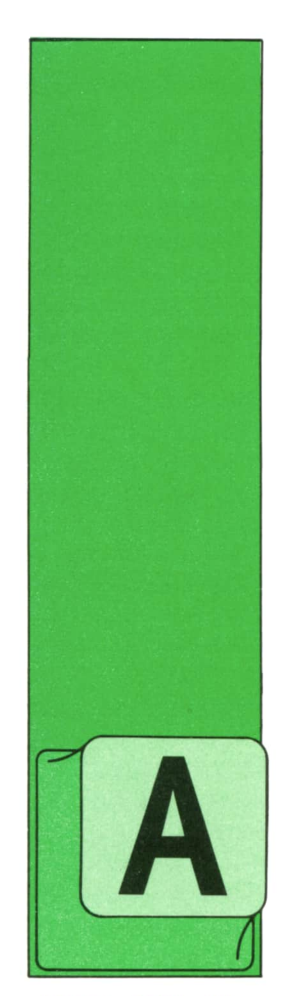


Como podemos ver, la posición de la tortuga no varía, ya que no le hemos dado ninguna orden para que se mueva.

Puedes probar tú a escribir lo que quieras. La tortuga te obedecerá.

Ahora tú puedes ir cambiando los colores del lápiz y de la pantalla e irle preguntando a la tortuga cuál es su número correspondiente. Podrás comprobar que nunca se equivoca.

PASCAL



L mostrar un número con el procedimiento WRITELN, se utiliza el espacio justo; por eso, si pusiéramos:

```
writein ('Peso',
    100 + 23, Ene);
```

en la pantalla saldría:

```
Peso 123–17
```

Esto se podría mejorar poniendo, por ejemplo:

```
writeln ('Peso ', 100 + 23,' ', Ene);
```

con lo que tendríamos:

```
Peso 123 –17
```

Sin embargo, es mejor indicarle al PASCAL de manera clara cuanto sitio queremos que ocupe un número. Esto se consigue poniendo justo a continuación de lo que se quiere enseñar el número de espacios a ocupar con dos puntos por delante; por ello:

```
writeln ('Peso', 100 + 23 :4, Ene:6);
```

daría:

```
Peso 123 –17
(    ••••       → 4 espacios para 123 )
(        ••••••  → 6 espacios para –17 )
```

Tras utilizar WRITELN, las siguientes cosas que queramos sacar aparecerán por fuerza en la siguiente línea (write significa escribir ln viene de line, línea). Habrá veces que, sin embargo, no queramos renunciar a la posibilidad de seguir escribiendo posteriormente en la misma línea; para esos casos está el procedimiento WRITE, que se utiliza exactamente igual que el anterior, pero sin forzar el paso a la siguiente línea.

Para aclarar todo esto, veamos el siguiente ejemplo:

```
program Ejemplo;

begin
   writeln ('Tras esto, nueva línea:');
   write ('Pero tras esto no: ');
   writeln ('¿lo ves?');
   writeln;
   writeln ('Y, por fin, me despido.')
end.
```

Tras compilar y ejecutar el programa, tendremos en la pantalla:

```
Tras esto, nueva línea:
Pero tras esto no: ¿lo ves?

Y, por fin, me despido.
```

En el programa se ha utilizado una vez el procedimiento WRITELN sin lista de cosas para sacar. Es como escribir «nada» pasándose luego, por supuesto, a la siguiente línea; por ello, queda un renglón vacío en la pantalla.

Dependiendo del compilador de que se disponga, existen otros procedimien-

tos adicionales para la salida de datos. Por ejemplo, para limpiar por completo la pantalla de lo que pudiera tener escrito previamente existe en algunos casos el procedimiento PAGE (página), que en otros se llama CLRSCR (de «clear screen», borrar pantalla). Para utilizarlo, bastaría con llamarlo por su nombre:

```
...
page;
writeln
('Esto sale en una pantalla limpia.');
...
```

Aunque menos, también es probable que se disponga de un procedimiento para poder escoger el sitio de la pantalla en que vaya a salir lo siguiente (algo así como el LOCATE del Basic). Por ejemplo, podría ser:

```
...
gotoxy (2 + 3, 17);
writeln ('Esto, en la línea 17, columna 5');
...
```

Esta posibilidad de variación de unos casos a otros se debe a que en el PASCAL estándar no hay nada indicado al respecto y, por tanto, está hecho a gusto de las personas que hayan desarrollado cada compilador. Cuando aprendamos a escribir nuestros propios procedimientos superaremos fácilmente todas estas diferencias.



Instrucciones de entrada

Al leer un número introducido por teclado como, por ejemplo, −17, y a la inversa de lo que sucede cuando queremos mostrarlo, hay que convertir esa secuencia de teclas en el código que utilizará internamente el ordenador para guardar ese número.

Por otra parte, los datos que se introducen desde teclado hay que guardarlos en memoria. Estos datos pueden ser distintos de una vez para otra, y, por tanto, habrá que guardarlos en una zona de memoria reservada para ello, es decir, en una variable.

Como es de suponer, existen procedimientos ya preparados que nos hacen todo esto. Los más corrientes son los denominados READ y READLN (read significa leer e ln, una vez más, viene de línea).

Normalmente, la única diferencia entre ambos estriba en que, cuando se utiliza READLN, una vez se ha llenado la variable con el dato tecleado, todo lo que se hubiera escrito en la misma línea hasta que se pulsó la tecla de RETURN (o ENTER o INTRO o NEWLINE según el ordenador) se pierde y no se puede aprovechar en una lectura posterior. Utilizándose READ, lo que resta en la línea queda disponible para las siguientes instrucciones de lectura.

(NOTA: Utilizando el TURBO PASCAL con el ordenador personal IBM o compatibles, para que READ funcione de esta manera es necesario poner tras la cabecera de programa, por ejemplo, (·$G128·), para así forzar la lectura de teclado a través del sistema operativo.)

Por ejemplo:

```
program LeeDatos;
var
  Peso            : 0..200;
  Letra1, Letra2 : char;

begin
  writeln
('Teclee dos letras (y pulse RETURN)');
  read (Letra1);
  readln (Letra2);
  writeln ('Han sido la ',Letra1,' y la',
           Letra 2, '¿verdad?');

  write ('Teclee el peso (y RETURN):');
  readln (Peso);
  write ('Le quedan sólo ',200 - Peso,
         ' para llegar a los 200.')
end.
```

Al utilizarse READ para Letra1, la segunda letra puede teclearse en la misma línea, sin separarla de la segunda pulsando la tecla de Return, pues seguirá disponible para su posterior lectura. Si se hubiera utilizado READLN para Letra1, la segunda letra se habría perdido y se tendría que volver a escribir (seguida de RETURN, que es la forma de indicarle al ordenador que ya se ha terminado de escribir el dato).

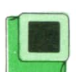
En el ejemplo también se hace la lectura de un número entero. Si el número tecleado fuese incorrecto (por ejemplo, X25) o estuviera fuera de rango (68030), el programa se pararía dando un aviso de error. En otra ocasión veremos cómo se puede corregir esto fácilmente.

Si varios datos van a ser leídos de una

sola vez, como sucede con Letra1 y Letra2, es posible hacerlo con una sola instrucción:

```
readln (Letra1,Letra2);
```

Esto equivaldría a las dos primeras instrucciones de lectura del ejemplo. Sin embargo, lo aconsejable es utilizar siempre una instrucción READLN por cada dato a leer, para así poder sacar por pantalla cada vez un mensaje indicativo de lo que hay que teclear.



Expresiones

Los dos primeros datos que se leían en el programa anterior simplemente se volvían a presentar sin hacer ningún proceso ni generar nuevos datos con ellos. Evidentemente, esto sólo sirve para muy poco. Mediante el uso de EXPRESIONES se pueden generar nuevos valores a partir de datos previos. Por ejemplo, a partir del peso tecleado hemos obtenido lo que queda para llegar a doscientos.

Una expresión es una lista de variables y constantes combinadas mediante OPERADORES para dar un resultado que, a su vez, puede ser guardado en una variable, mostrado en la pantalla o impresora, etc.

En una expresión SOLO se pueden combinar constantes y variables de un mismo tipo. No tiene sentido sumar un número a una letra. Por ello, según sea el tipo de los elementos que la integran, se dice que una expresión es de tal tipo o de tal otro.

En principio, hay cinco operadores que se pueden utilizar en expresiones de tipo INTEGER. Se escriben como +, –, •, DIV y MOD:

+ y – sirven para sumar o restar los valores que se encuentran por delante y detrás de ellos: 3 + 1 hace que se sumen 3 y 1 y, por tanto, equivale a poner 4.

La multiplicación se indica con •: 2 • 3 es igual a 6.

DIV representa la división entera, es decir, sin obtener decimales. 5 DIV 2 da como resultado 2, al igual que 4 DIV 2.

MOD es el resto de la división entera. 5 MOD 2 vale 1, mientras que 4 MOD 2 da 0.

Las expresiones de tipo INTEGER se calculan empezando por la izquierda, pero realizándose las sumas y restas después de las demás operaciones. Por ejemplo:

```
5 –4 + 9 MOD 5 + 7 • 3 primero • y MOD:
5 –4 +         4 + 21   luego + y –:
  1  +         4 + 21
     5           + 21
                26
```

Sin embargo, se puede alterar el orden de cálculo poniendo entre paréntesis lo que se desea calcular en primer lugar:

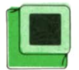
```
5 – (4 + 9) MOD (5 + 7) • 3  primero lo
                             de los
                             paréntesis:
5 –      13 MOD     12  • 3  luego •
                             y MOD:
5 –            1        • 3
5 –                     3
  2
```

Se pueden utilizar todos los paréntesis que se quiera e incluso paréntesis dentro de paréntesis. Es una buena práctica poner paréntesis en cuanto haya la más mínima duda sobre el orden de cálculo de una expresión, pues además la expresión queda mucho más clara, pero conviene vigilar que todos los paréntesis que se abran se cierren luego, pues si no se producirá un error al compilar el programa.

El operador – aplicado a un valor aislado sirve para cambiar su signo; se dice en este caso que es un operador «monádico», mientras que en el caso normal es un operador «diádico» (es decir, se aplica a dos valores):

```
2 • ( – (5 + 4 • 3)) s igual a
2 • (–17) y esto es igual a –34.
```

Una expresión de un tipo dado se puede poner en cualquier lugar en que pueda ir una variable o constante del mismo tipo, salvo advertencia en contra, y equivale a poner su resultado en el momento de hacer el cálculo.



Asignación

Ya hemos visto cómo, mediante los procedimientos READ y READLN, es posible guardar datos tecleados en variables. Sin embargo, las variables se utilizan para guardar no sólo datos tecleados, sino datos elaborados por el propio programa. Esto se consigue mediante las instrucciones de asignación.

Estas constan de dos partes separadas por el operador de asignación. A la izquierda se escribe el nombre o identificador de la variable en que se desea guardar el dato y a la derecha la constante, variable o expresión cuyo valor se desea guardar.

El operador se escribe como :=, o sea, dos puntos INMEDIATAMENTE seguidos de un signo igual. En definitiva:

Nombre de la variable := expresión, constante o variable

El valor a guardar DEBE ser del mismo tipo que la variable en que se guarda. No se pueden mezclar nunca tipos distintos. Veamos un ejemplo:

```
program Asignacion;

const
   Dos = 2;
var
   N: integer;
   C: char;

begin
   N:= 7;
   C:= 'A';
   writeln ('N vale ',N,' y C vale ',C);
   N:=N+1;
   writeln ('N vale ahora ',N);

   N:= N div Dos + 2;
   writeln ('N vale ahora ',N)
end.
```

Al correr el programa, tendremos en la pantalla:

```
N vale 7 y C vale A
N vale ahora 8
N vale ahora 6
```

Está claro que para guardar un valor en una variable antes hay que calcularlo. Por ello, en los casos en que la variable a la que vamos a asignar el valor aparece también a la derecha, se utiliza su valor anterior para obtener el resultado de la expresión, y sólo una vez calculada ésta pasará la variable a tener el nuevo.

Así, la instrucción N:=N + 1 toma el valor anterior (7 en el ejemplo) para calcular la expresión. Una vez hecho esto, el resultado (8 en el ejemplo) se guarda en N.

Si a continuación hubiera otra instrucción N:=N + 1, N pasaría a valer 9, etc.

OTROS LENGUAJES

SISTEMA OPERATIVO: MS-DOS

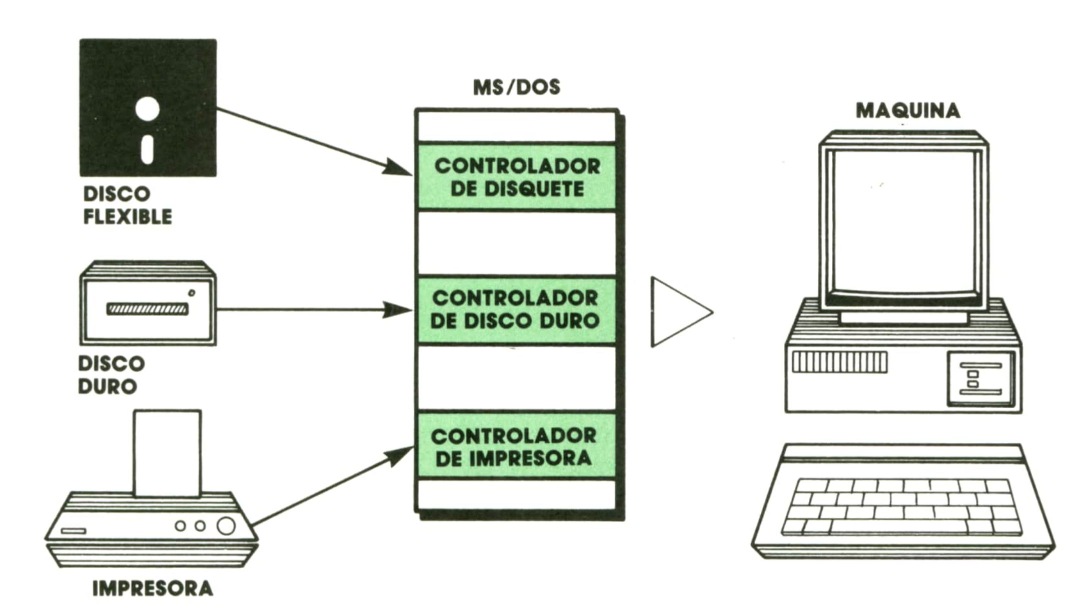


Introducción



S/DOS significa Microsoft-Disk Operating System. Es un sistema operativo creado por la compañía norteamericana Microsoft para ordenadores basados en los microprocesadores 8088 y 8086 de Intel. Antes de que IBM lo adoptase como sistema operativo para su ordenador personal, pugnaba con el CP/M 86 por convertirse en un estándar en el terreno de los equipos de 16 bits, pero a raíz de que la poderosa multinacional norteamericana lo adoptase para su IBM PC, MS/DOS se ha convertido en el rey indiscutible de los sistemas operativos para equipos basados en un microprocesador de 16 bits.

Conceptualmente, MS/DOS, al haber seguido las pautas de CP/M, guarda gran similitud con éste, pero también tiene muchos de sus defectos. MS/DOS está orientado principalmente a la gestión de archivos (o ficheros) en disco. En su primera versión, DOS 1.0 está configurado para operar con 2 unidades de disco flexible, mientras que la versión 2.0 (y posteriores) viene preparada para controlar unidades de disco rígido.





En MS/DOS los dispositivos periféricos están controlados por módulos independientes.

Debido a que las primeras versiones de MS/DOS se encuentran en la actualidad completamente desfasadas y superadas, aquí nos referiremos a las versiones desde la 2.00 en adelante, aunque muchos conceptos son válidos para todas ellas.

En MS/DOS, cada uno de los distintos dispositivos periféricos conectados al ordenador tiene un programa o módulo propio del sistema operativo encargado de controlarlo. Esto constituye una gran ventaja, ya que el equipo puede adaptarse al uso de todo tipo de periféricos con sólo añadir un módulo que controle cada nuevo dispositivo conectado; pero a la vez es también un inconveniente, ya que al estar pensado el sistema operativo en función de las necesidades del ordenador y no de las del usuario, obliga a éste a conocer de manera precisa las distintas órdenes del MS/DOS.



Ficheros en MS/DOS

Un fichero es un conjunto de información al que se le asigna un nombre, y que está guardado en un dispositivo de almacenamiento secundario, como puede ser un disco flexible o un disco duro.

La información que contiene un fichero es muy variada (programas, texto, datos...) y lo más normal es que sea homogénea, es decir, lo usual es que un fichero contenga un programa, o un texto, o datos de alguna aplicación, pero no cosas mezcladas.

Los ficheros se referencian por su identificación, que está compuesta, de forma obligatoria, por un nombre, y opcionalmente por una extensión, separadas ambas partes por un punto. El nombre debe estar formado por al menos un carácter, siendo su longitud máxima de 8. Estos pueden ser letras, números o determinados signos de puntuación que no se utilicen en la formulación de comandos. Por ello, no se admite la inclusión de puntos ni espacios en blanco dentro del nombre.

La extensión del nombre del fichero es una cadena, de, como máximo, 3 caracteres. Se utiliza para que el usuario tenga una idea del tipo de información que contiene el fichero. Por ejemplo, los comandos externos del MS/DOS residen en ficheros cuya extensión es ".COM" (para indicar que son COMandos). Un programa en lenguaje BASIC podría guardarse en un fichero con extensión ".BAS", etc.

```
 Volumen en unidad A sin etiqueta
 Directorio de  A:\

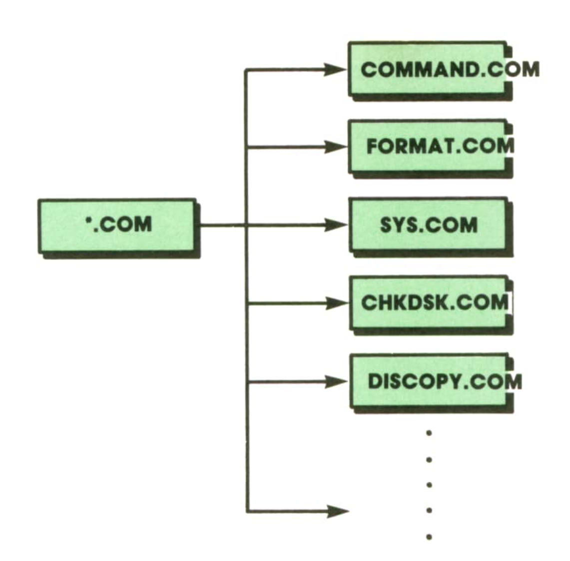
COMMAND  COM    23898  22/04/85  12:00
CONFIG   SYS       14  22/04/85  12:00
AUTOEXEC BAT       41  22/04/85  12:00
ANSI     SYS     1651  22/04/85  12:00
ASSIGN   COM     1516  22/04/85  12:00
ATTRIB   EXE    15107  22/04/85  12:00
BACKUP   COM     5844  22/04/85  12:00
BASIC    COM    17792  22/04/85  12:00
BASICA   COM    27520  22/04/85  12:00
CHKDSK   COM     9867  22/04/85  12:00
COMP     COM     3826  22/04/85  12:00
DISKCOMP COM     4201  22/04/85  12:00
DISKCOPY COM     4489  22/04/85  12:00
EDLIN    COM     7421  22/04/85  12:00
FDISK    COM     8381  22/04/85  12:00
FIND     EXE     6436  22/04/85  12:00
FORMAT   COM     9603  22/04/85  12:00
GRAFTABL COM     1190  22/04/85  12:00
GRAPHICS COM     3111  22/04/85  12:00
JOIN     EXE    16019  22/04/85  12:00
KEYBFR   COM     2473  22/04/85  12:00
KEYBGR   COM     2418  22/04/85  12:00
KEYBIT   COM     2361  22/04/85  12:00
KEYBSP   COM     2451  22/04/85  12:00
KEYBUK   COM     2348  22/04/85  12:00
LABEL    COM     1918  22/04/85  12:00
MODE     COM     5434  22/04/85  12:00
MORE     COM      286  22/04/85  12:00
PRINT    COM     8595  22/04/85  12:00
RECOVER  COM     4194  22/04/85  12:00
RESTORE  COM     5667  22/04/85  12:00
SELECT   COM     2107  22/04/85  12:00
SHARE    EXE     8320  22/04/85  12:00
SORT     EXE     1664  22/04/85  12:00
SUBST    EXE    16643  22/04/85  12:00
SYS      COM     3839  22/04/85  12:00
TREE     COM     2877  22/04/85  12:00
VDISK    SYS     3398  22/04/85  12:00
DEBUG    COM    15638  22/04/85  12:00
LINK     EXE    38144  22/04/85  12:00
40 Archivo(s)            0 bytes libres
```



Listado del directorio del disco del sistema operativo.

Un tipo de ficheros algo diferentes a los demás son los denominados de tipo "batch". En éstos, el usuario almacena una lista de órdenes o comandos de MS/DOS, que se ejecutan de manera secuencial desde el momento en que el usuario manda al sistema operativo que procese el correspondiente fichero batch. Estos ficheros, cuya utilidad es ahorrar trabajo y tiempo al usuario, tienen la extensión ".BAT". De este modo, si tenemos una serie de órdenes que utilizamos de manera repetida, se evita el tener que teclearlas una por una cada vez que se necesite emplearlas.

La flexibilidad en el manejo de estos fi-




Con el uso de caracteres comodín podemos nombrar varios ficheros a la vez.

cheros viene dada por las siguientes características:

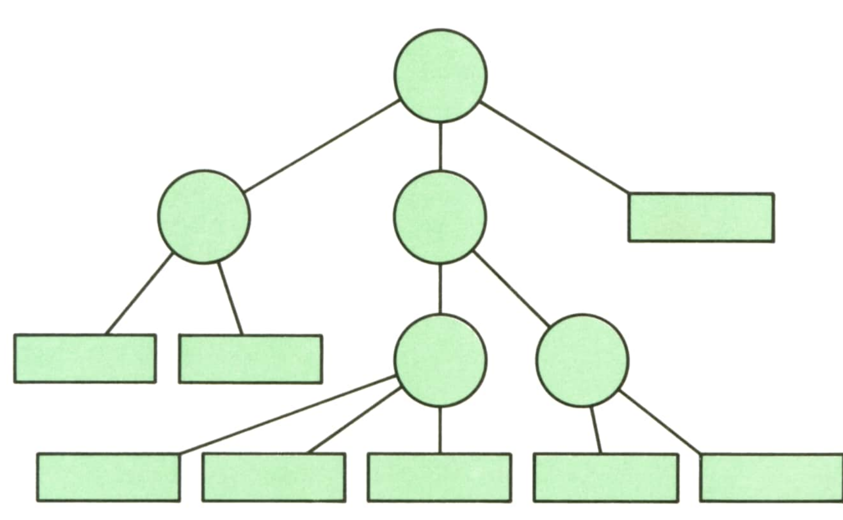
— La existencia de unas "órdenes internas" o subcomandos, que permiten al usuario crear un "programa de comandos MS/DOS" dentro del fichero batch. Es posible, por ejemplo, ejecutar iterativamente un grupo de órdenes, seleccionar los mandatos a procesar, esperar la entrada desde el teclado, etc.

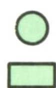
— Al crear un fichero batch se pueden dejar comandos con parámetros variables cuyo contenido se especificará en la orden de procesamiento del fichero batch.

— Enlace de ficheros batch. Aunque de forma restringida, desde un fichero batch se puede provocar la ejecución de otro. La única limitación es que las llamadas deben colocarse al final del fichero.

El manejo de los ficheros en MS/DOS se facilita por una serie de "utilidades" que se pueden considerar "estándar", pues existen en la mayoría de los sistemas operativos. Una de ellas es la referenciación de ficheros mediante el uso de caracteres "comodín". Su utilidad reside en que permiten extender la actuación de un comando a un grupo genérico de ficheros, sin necesidad de especificar el nombre de cada uno de ellos.

En MS/DOS existen dos caracteres "comodín": el asterisco ("*"), que se utiliza para sustituir un grupo de caracteres, y la interrogación ("?"), para sustituir un solo carácter. Por ejemplo, si queremos referenciar a todos los ficheros de extensión ".COM" del disco del sistema, no es necesario nombrarlos todos, sino que basta con poner "*.COM". El ordenador entiende que es una referencia a todos lo ficheros cuyo nombre sea cualquier cadena de caracteres (esa es la interpretación del "*"), seguida de la cadena ".COM". De igual manera, si tuviéramos tres ficheros de nombre "CAP1.TEX", "CAP2.TEX" y "CAP3.TEX", podríamos referenciarlos a todos poniendo "CAP?.TEX".






La jerarquización de los directorios nos permite tener de una manera clara y ordenada todos los datos.



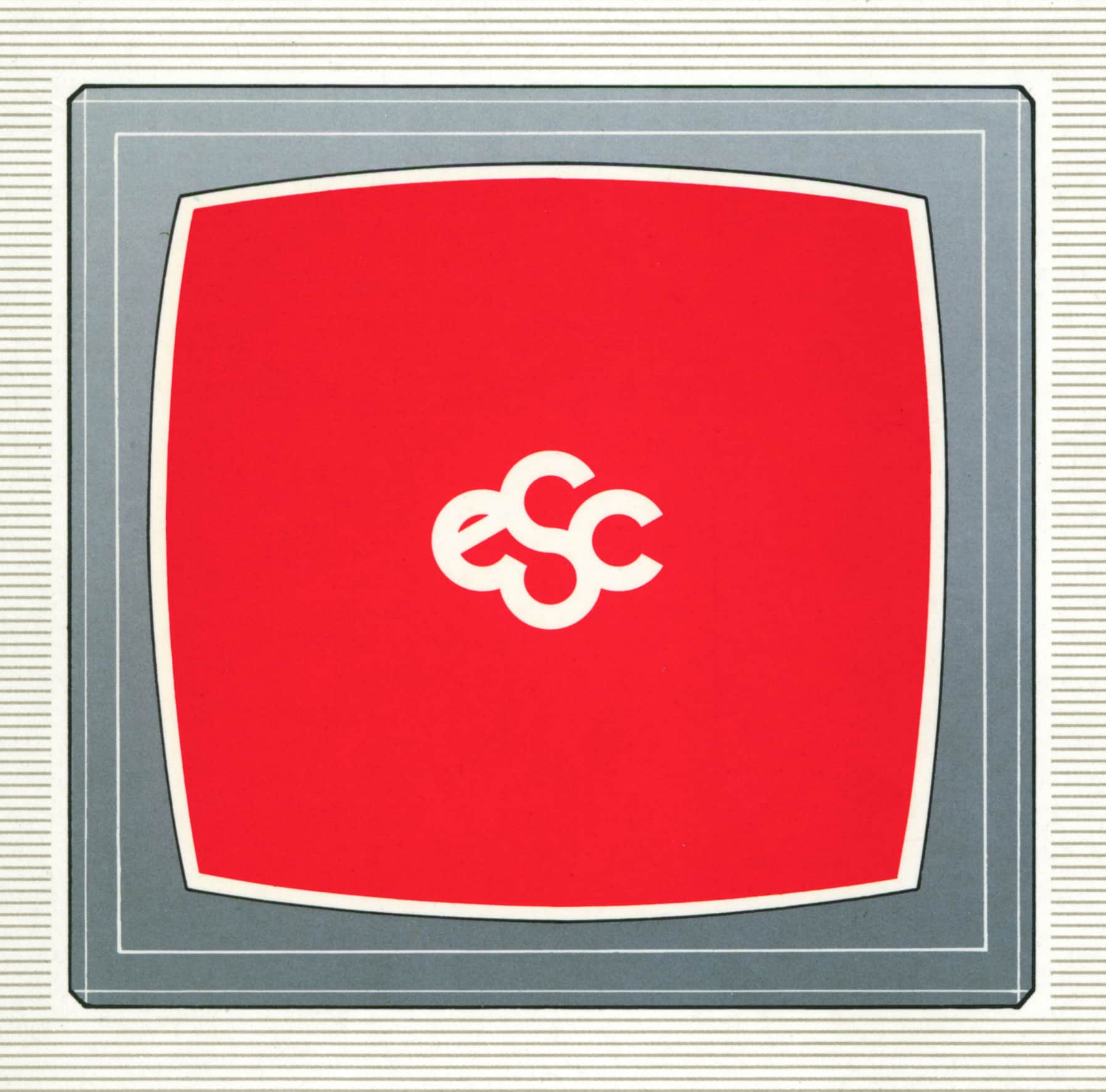
Organización de ficheros en MS/DOS

Una de las novedades que aportó la versión 2.00 de MS/DOS es la organización de los ficheros en directorios. Un directorio es una colección de ficheros o a su vez de otros directorios (también llamados subdirectorios). Estos subdirectorios pueden contener más ficheros y/o subdirectorios, desarrollándose así una estructura ramificada en forma de árbol invertido, cuya fuente o inicio es el directorio raíz, identificado por el carácter "\".

La utilidad de este sistema es que se pueden estructurar los datos que se refieren a un tema específico en su propio directorio, contribuyendo así a tener una estructura clara y ordenada de los mismos.

Si trabajando en un subdirectorio se quisiera hacer referencia a un fichero de otro subdirectorio, al nombre de este fichero habría que anteponer la descripción de la rama a la que pertenece. Esta descripción es lo que se conoce como "path" o "camino" de un fichero. Este camino puede ser absoluto o completo (si se incluye la descripción de todos los niveles o subdirectorios de la rama, empezando en el directorio raíz), o relativo (si se parte desde el subdirectorio en el que se está trabajando).

MS/DOS pone a disposición del usuario comandos para realizar casi cualquier operación relativa a los directorios, tales como creación, borrado, listado de su contenido, movimiento a través de la estructura en árbol, etc.



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼